NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

THE NAVIGATION DATA LOGGER
FOR
A SUITCASE NAVIGATION SYSTEM

by

CHIN, YU-CHI
June 1991

Thesis Advisor: Dr. Uno R. Kodres

Approved for public release: distribution is unlimited.

T256813

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): CS	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
THE NAVIGATION DATA LOGGER FOR A SUITCASE NAVIGATION SYSTEM

12. PERSONAL AUTHOR(S)
CHIN, YU-CHI

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM 9/90 TO 06/91	JUNE 1991	66

16. SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
A small navigation data logger software system

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis presents the design, implementation and description of a Data_Logger for the Suitcase Navigation System. All the programs and examples presented in this thesis were implemented in the Ada programming language, which has sucessfully incorporated the low_level I/O ports communication with the high_level abstraction. The software is portable as desired and can be reused by LCCDS when needed.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr. Uno R. Kodres	22b. TELEPHONE (Include Area Code): (408) 646-2197; 22c. OFFICE SYMBOL: 52Kr

Approved for public release; distribution is unlimited

THE NAVIGATION DATA LOGGER
FOR
A SUITCASE NAVIGATION SYSTEM

by
Chin, Yu-Chi
Commander R. O. C. NAVY
B. S. Chinese Naval Academy, 1977

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
JUNE, 1991

Robert B. McGhee, Chairman,
Department of Computer Science

THE NAVIGATION DATA_LOGGER
FOR A SUITCASE NAVIGATION SYSTEM

Author : Chin, Yu- Chi
Commander, R. O. C. (Taiwan) Navy
B. S., Chinese Naval Academy, 1977

ABSTRACT

This thesis presents the design, implementation and description of a Data_Logger for the Suitcase Navigation System. All the programs and examples presented in this thesis were implemented in the Ada programming language, which has sucessfully incorported the low_level I/O ports communication with high_level abstraction. The software is portable as desired and can be reused by LCCDS when needed.

C4447
C.1

•Acknowledgments

To my thesis advisor, Dr. Uno R. Kodres. I would like to express my sincere thank for all the confidence and support, which has never stopped during past three quarters. The knowledge I've learned and the joy of work done we shared, was the words for those days. I know that I've learned a lot from the best.

Also, I want to send a special thanks to the friends here in the school, the friends we always work together, and the friends whom always tried their best to help. Direct help from Pat Barnes, Jeff, John, Russ, and all the technical staff in the CS Department, Thank you all very much.

To my family, I want to dedicated this thesis to the big family support, especially to my dear wife Anne (Chin/Chang, Huei-Yen), our children Lanny (Chin, Lan-Ting) and Lanchun (Chin, Lan-Chun). Most importantly, Thanks God.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. BACKGROUND 1

B. PURPOSE OF THIS THESIS 1

C. THESIS ORGANIZATION 1

II. SUITCASE NAVIGATION SYSTEM OVERVIEW 3

A. INITIAL PROBLEM STATEMENT 3

1. The Suitcase Navigation System Project 3

2. The Physical System 3

B. SYSTEM DESCRIPTION 4

C. SYSTEM OPERATIONS 5

1. CRT display 5

2. Plotter 5

3. Data Logger 5

4. Remote CRT 6

5. Keyboard 6

D. REQUIREMENTS ANALYSIS 6

1. The Statement of Purpose 6

2. Suitcase Navigation System Context Diagram 6

3. Suitcase Navigation System Level-0 DFD 7

III. DATA_LOGGER DESIGN 8

A. INTRODUCTION 8

B. DEFINE THE PROBLEM 9

C. IDENTIFY THE OBJECTS 10

D. IDENTIFY THE OPERATIONS 10

1. Operation 1: INITIALIZATION--Set up external Nav_sensors connection 10

2. Operation 2: External Nav_Sensor operation setup 11

3. Operation 3: LOGGER--Log and Convert 11

4. Operation 4: DATA_LOGGER-- the main program unit 11

5. Operation 5: POSITION_NOW--Display current data string 12

E. ESTABLISH THE COMMUNICATION 12

F. ESTABLISH THE INTERFACE 14

IV. DATA_LOGGER IMPLEMENTATION 16
A. SYSTEM-SENSOR INTERFACING 16
Code-1: Initialization 16
Code-2: Setup the Baud_Rate 17
B. SENSOR DATA HANDLER 18
Code-3: A main subprogram 19
C. SYSTEM DATA MANAGEMENT 21
Code-4: Display current position data 21
D. PROGRAM CODING 22
V. DESCRIPTION OF THE DATA_LOGGER 29
A. THE REQUIRED PROGRAM STRUCTURE 29
1. Hardware Structure 29
2. Software Structure 30
B. ADVANTAGES OF THE DATA_LOGGER 30
C. USER's MANUAL 31
VI. CONCLUSIONS AND RECOMMENDATIONS 32
A. CONCLUSIONS 32
B. RECOMMENDED FOLLOW-ON WORK 32
APPENDIX--I GPS Trimble-4000S 34
APPENDIX--II Predefined PACKAGE BIT 37
APPENDIX--III SAMPLE DATA 38
APPENDIX--IV Ada Program Examples 43
APPENDIX--V: USER's MANUAL (Operations and Responses) 51
APPENDIX--VI: TEXT NOTES FROM GRADY BOOCH 54
LIST OF REFERENES 56
INITIAL DISTRIBUTION LIST 57

LIST OF FIGURES

FIGURE-1: Basic Physical System.....5

FIGURE-2: Suitcase Navigation System Context Diagram.....6

FIGURE-3: Level Zero Data Flow Diagram.....7

FIGURE-4: Process Nav_Sensor Data.....8

FIGURE-5: The Environmental I/O Problems.....9

FIGURE-6: Description of Logger Design.....13

FIGURE-7: Description of Data_Logger Design.....14

FIGURE-8: Description of Data_Logger I/O ports.....15

FIGURE-9: Hardware Structure of Data_Logger.....29

I. INTRODUCTION

A. BACKGROUND

The research interests of the Small Navigation Data Logger System was initially sponsored by the Pacific Missile Test Center (PMTC), who uses various ships at their site and other locations around the world. They often need to have the navigation data from the system that the particular ship uses logged and displayed. This system can be one of many. Accordingly PMTC desires a small, portable, data logger that will accept the inputs from any of several navigation systems. It will have to interface with only one system at once. The system should perform the function of "Accepting the data, logging the data, displaying the data graphically, plotting the data". Real time navigation will often be done from this system.

There are several systems that either exist or are being proposed to perform functions like this. Most are expensive, and require large amount of memory. In addition commercial systems do not provide the source code or other methods for easy upgrades for new systems. Some of the reasons for the large memory size of these systems is the desire to interface with different electrical standards. Our Suitcase Navigation System is concerned only with the RS-232 electrical standardinterface.

B. PURPOSE OF THIS THESIS

The objective of this thesis is to provide a generic system, based on the Ada language, to create an interface to the Suitcase Navigation System, which accepts positional data from a class of Global Positioning Systems (GPS) and distributes this information to a variety users including the LCCDS's navigational subsystem. The central focus of the thesis is to design, implement, and experience to gather performance data of the Navigational Data_Logger System.

C. THESIS ORGANIZATION

Chapter II gives the Suitcase Navigation System overview. In order to assist the reader

in understanding the basic features of SNS, which follows the Yourdon's Modern Structured Analysis 1989 (Reference-2), to present a software develop method from the Requirements Analysis point of view. The Suitcase Navigation System's Context Diagram and level-0 Data Flow Diagram is presented in this chapter.

Chapter III, the object-oriented method suggested in Grady Booch's Software Engineering with Ada (Reference-3), is applied to the design of Navigation Data_Logger. In order to describe to the reader parts of the system function, the chapter defines the problem, identifies the objects, identifies the operations, establishes the communication, and establishes the interface.

Chapter IV describes the implementation of the Navigation Data_Logger, the System-Sensor Interface, the Sensor_Data Handler, The System Data Management, and program coding.

Chapter V describes the performance of the Data_Logger. We start from a required program structure, which described hardware structure and software structure. The simulation in our lab, provides experimental results for Hardware connection, Transmission Rate, and Conversion accuracy, we describe the working process, and also point out the effects and contacts. Then from the user's point of view we describes how to use the Data_Logger.

Chapter VI is the final chapter, which includes the Conclusions and Recommendations of some suggestions for future research and follow-on work.

Appendix.1: The specification of the RS-232 connections between external Navigation_ Sensor and the working system.
Appendix.2: Predefined Ada package Bit specification.
Appendix.3: Sample data formats from external navigation sensor.
Appendix.4: Sample Ada programs for the project development.
Appendix.5: Navigation Data_Logger User's manual.
Appendix.6: Text Notes from Grady Booch.

II. SUITCASE NAVIGATION SYSTEM OVERVIEW

A. INITIAL PROBLEM STATEMENT

1. The Suitcase Navigation System Project

There is a need for an integrated navigation system for use with various ships or aircraft of opportunity. System must be self-contained, lightweight, portable, and multi-purpose. Systems should use off-the shelf components, to facilitate hardware replacement and minimize downtime. System would be utilized for the following tasks:

> "Integrated Navigation, Tracking, and Positioning. Test and Evaluation of Equipment, System, and Weapons. Fleet Training Umpire. Range Surveying."

System must be able to accept input data from various navigational sources. It will provide real-time processing of data and in the absence of data will use dead reckoning points derived from a Kalman Filter. It will allow output data to be displayed on CRT, and XY plotter.

2. The Physical System

1) Basic System: Computer (Laptop preferred), Keyboard, and CRT.
2) Accessories: Printer, UHF Modem, Remote CRT, and XY plotter.
3) Allowance for selection from the following input systems (initially desired are):
- TI 4100 GPS Receiver
- Trimble GPS Receiver
- IEC GPS Receiver
- Motorola GPS Receiver
- Magellan GPS Receiver
- Rockwell-Collins (JPO) GPS Receiver
- Mini-ranger Radio Positioning System

4) Allowance for selection from the following output systems

- Data recorded by:

a. Disk, floppy or hard drive.

b. Tape.

c. Cassette.

d. Printer.

- Visually displayed by:

a. CRT.

b. Printer.

c. XY Plotter.

d. Remote CRTs connected via UHF modem (for use on flight station, bridge, or deck).

B. SYSTEM DESCRIPTION

System is turned on. Software must handle initial setup accessing I/O ports, through the hardware bus. CRT displays setup menu, requires keyboard entries. Requests the following:

- Mode type: provide external setup on navigation input systems.
- Type of input: primary, secondary, and tertiary navigation source and its format.
- XY plotter: plot functions.
- Navigational data recording: utilization, location, and rate.
- Printer recording: utilization, format, and rate.
- Remote CRT: utilization and setup.
- Verification of entries: user friendly design.

Figure-1 Basic Physical System of Suitcase Navigation System illustrates the example of the Suitcase Navigation System that PMTC proposed to the Naval Postgraduate School for the purpose of:

"To provide PACMISRANFAC/ PACMISTESTCEN with an integrated navigation system capability. This capability would provide to range users, a means to pre-plan range operations and to accurately navigate, position, and collect operational data. In addition, this capability may be the make or break point between a range user using or not using the range."

-- Note: A work statement from PMTC.

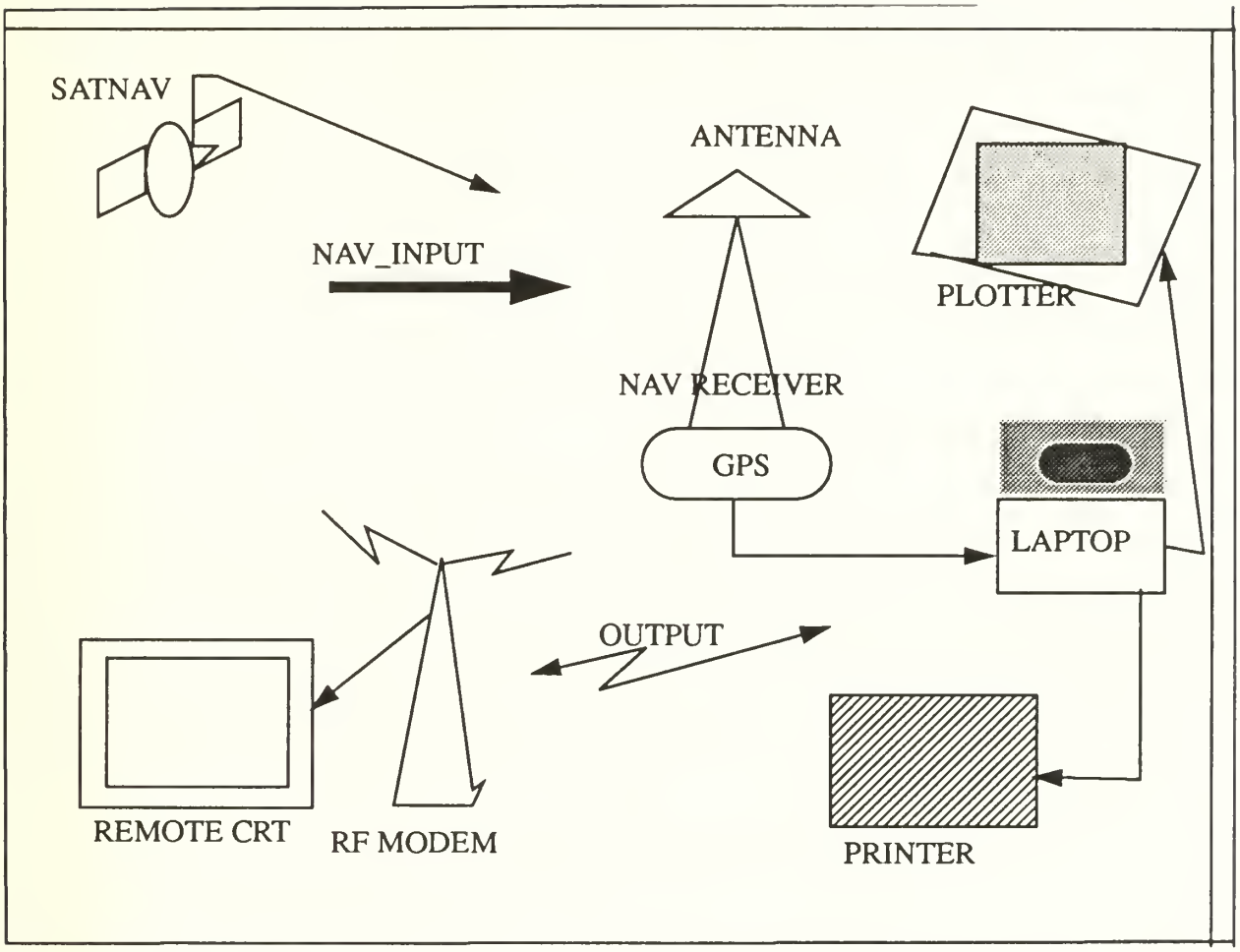

Figure-1: Basic Physical System

C. SYSTEM OPERATIONS

1. CRT display

Two-thirds of screen for graphic display and switchable provides: "duplicate of XY plotter, Positional Bulls-eye-Range and Bearing, Left/Right of Track-Waypoints". Remainder of screen provides "Data and Time, Latitude and Longitude, Range and Bearing for Bulls-eye and Waypoints, Help notations for program change toggles, Edge of plot sheet warning".
-- Statements from "World Range Suitcase Navigation Feasibility Study". (Reference-4)

2. Plotter

Responds as prompted from setup and/or program change toggles.

3. Data Logger

Responds as prompted from setup and/or program change toggles.

4. Remote CRT

Displays as prompted from setup and/or program change toggles.

5. Keyboard

Inputs system setup and provides program change toggles.

D. REQUIREMENTS ANALYSIS

1. The Statement of Purpose

The purpose of the Suitcase Navigation System (SNS) is to generate a general purpose Ada language based system, which takes inputs from different Global Positioning Systems, and record position data generated by any of the positioning system via an RS-232 interface. The general purpose software would be flexible and programmable by selecting components off the menu presented to the screen of the lop-top Any graphics capability of the laptop could be used to present the geographic position of the platform in an appropriate geometric design. (Reference-4)

2. Suitcase Navigation System Context Diagram

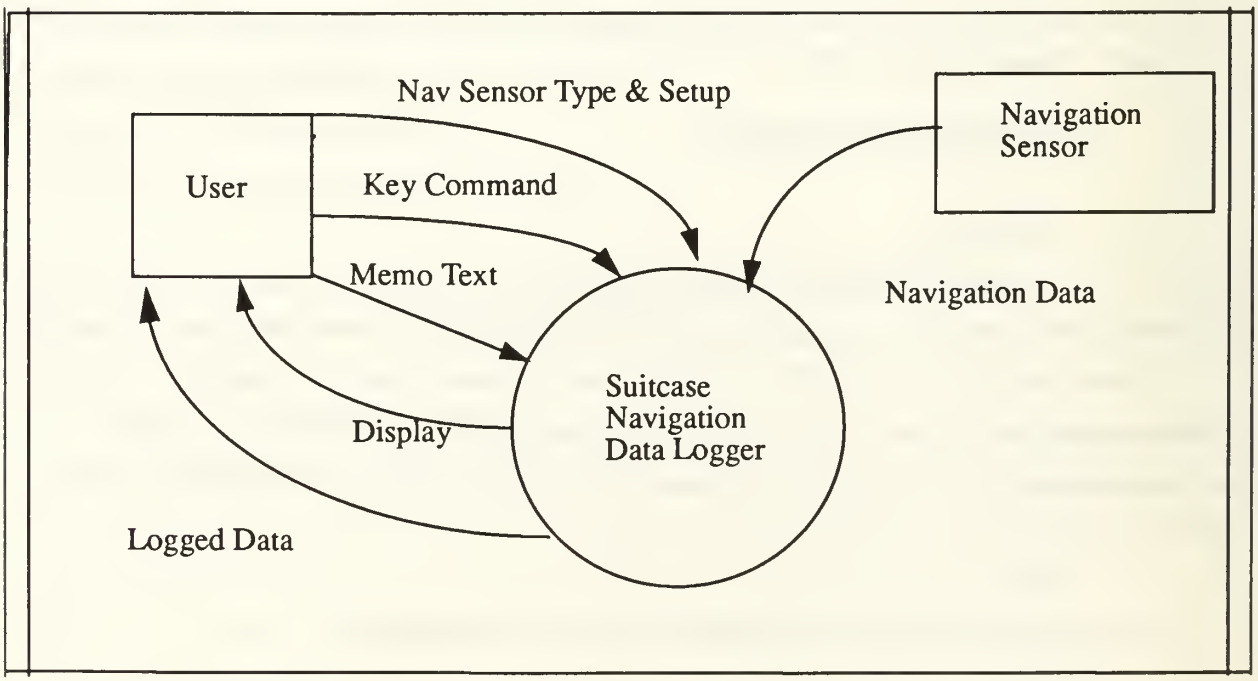

Figure-2: Suitcase Navigation System Context Diagram

3. Suitcase Navigation System Level-0 DFD

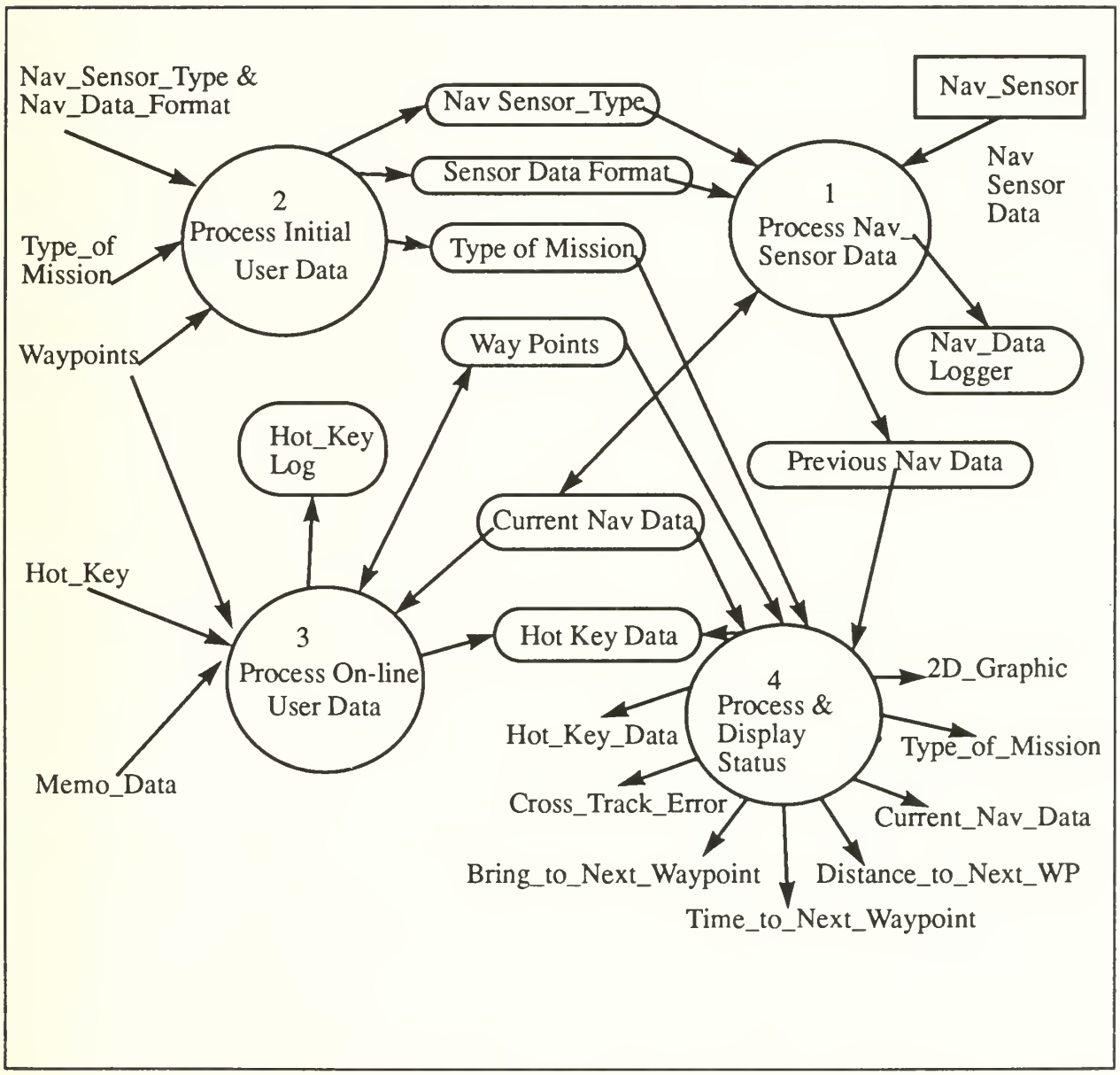

Figure-3: Suitcase Navigation System Level-0 DFD

III. DATA_LOGGER DESIGN

A. INTRODUCTION

The system is designed for the Suitcase Navigation System project. To implement Global Positioning System (GPS) data as the resource to satisfy the navigation problem needs, and using the highly accurate position to develop the whole system as a reliable console by means of Man Machine Interface. This thesis is look into the first bubble of the level-0 Data Flow Diagram, which interface the external nav_sensor, and process the nav_sensor data. Figure-4 illustrates the design.

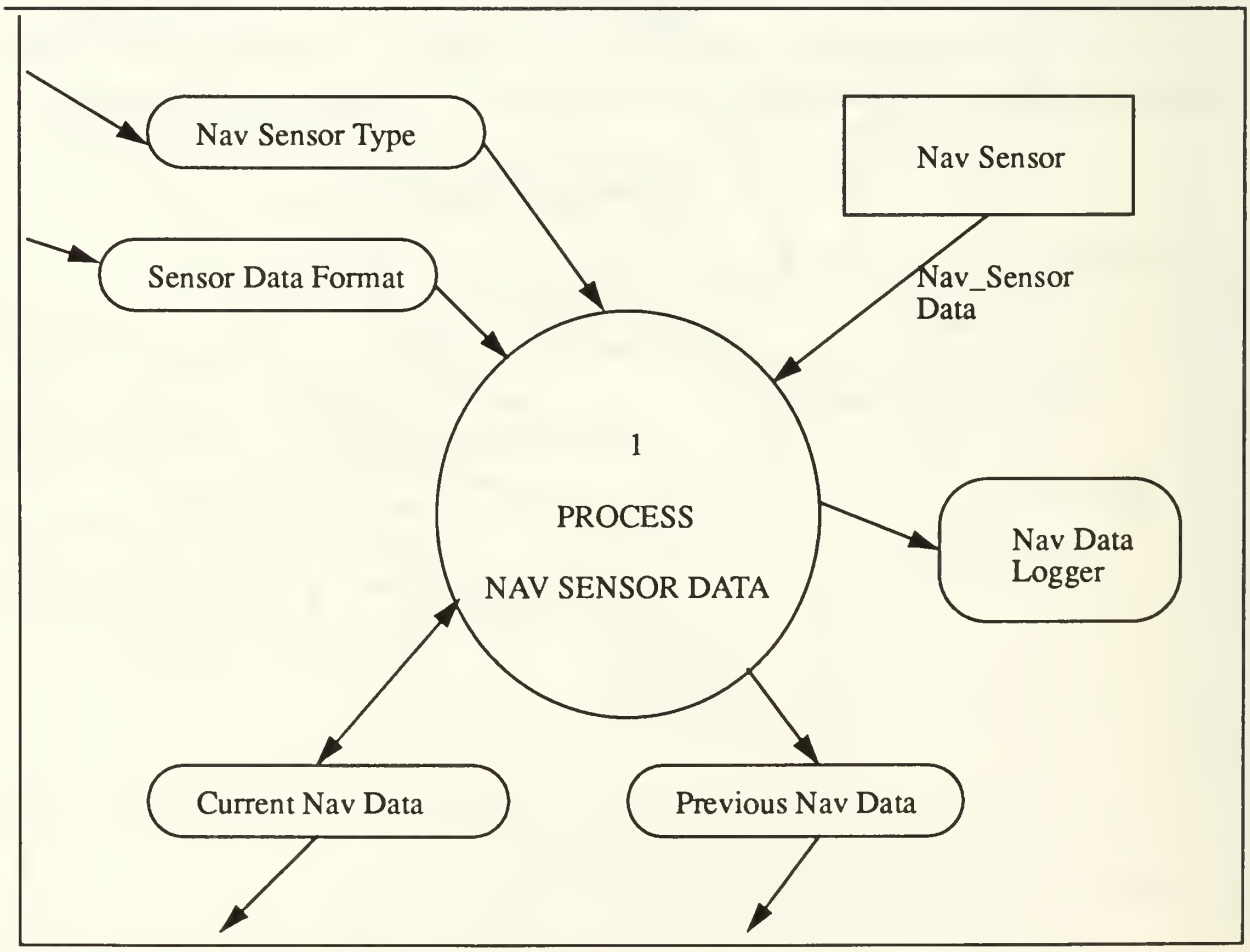

Figure-4: Process Nav_Sensor Data

B. DEFINE THE PROBLEM

The primary concern is the monitoring or control of real-time processes. Figure-5. The environmental I/O problem (Process Data Received From Sensor), illustrates the problem space. Our problem does involve several navigation sensors in various modes of operation. These sensors continuously sample the ambient data and transmit in several different rates. If a particular set of sensor data which is needed by the user, the Data_Logger shall be able to get and print it. The system also displays the data while it is logging or printing. We are presently using Trimble-4000s GPS as the external Nav_sensor, and a laptop computer as the hardware system.

To process nav_sensor data, the system need not only log the data, but allow the user to interact with the system by exception handling design (Hot_Key). Periodically, our system must be able to get the current data_string of navigation sensor to a permanent log. This can be done by pulling data out from the buffer file and print it. Printers have the nasty habit of running out of paper at the worst possible time; therefore, we have judged the printer to be the least reliable device in our system.

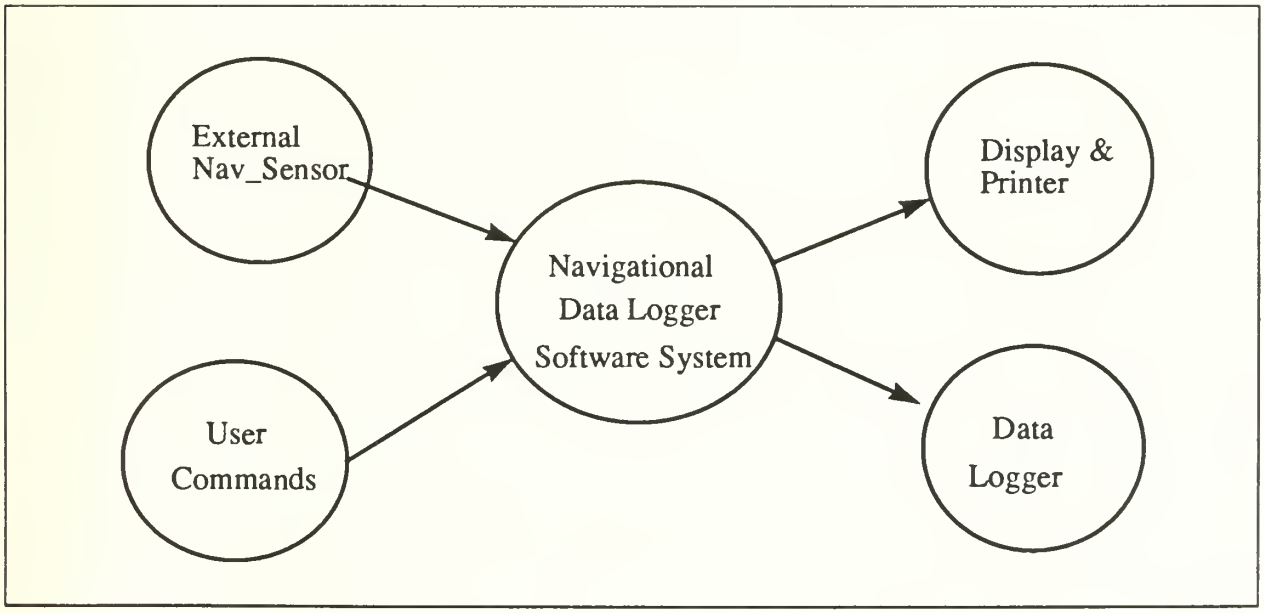

Figure-5: Environmental I/O problems

C. IDENTIFY THE OBJECTS

- External Nav_Sensors (GPS).
- CRT and Printer.
- Data_Logger.
- User_Commands.

The external Nav_Sensors defines a generic class of objects, whereas our system target is an instance of an object using the Trimble-4000 S GPS.

The CRT and Printer defines another set of external devices which act as a servers in the system.

Data logger defines the flow of data transfer, which identified the legal character from line ports, converts it and stores it. The data is logging to raw_data file, buffer file and display.

A user exists outside the design application, but the user can interact with the system via commands. A software development of hot key design will be implement in the future.

D. IDENTIFY THE OPERATIONS

To consider the behavior of each object, we shall specify the operations and identify the concurrence. According to Booch's concurrent real-time processing, we shall abstract the External Nav_Sensor as a concurrent entity. It's primary role is to continuously monitor the input data. CRT and Printer as server of Data_Logger, and Data_Logger as the actor of the software system. From the constrains of software support in Suitcase Navigation System, which assigned Laptop as the working system and DOS as the operating system, our project starts from the basic process operations that the DOS can support and upgrade in the future.

The Operations of the Navigation Data_Logger is organized as follows:

1. Operation 1: INITIALIZATION--Set up external Nav_sensors connection
   *. Assign communication port: COM1 or COM2;
   -- GET(NUM): Select proper com_port and assign variable value;
   *.Assign transmission rate: Baud_Rate
   -- GET(CH): Match the initial selection;
   -- OUTPORT (P1, 128): Access baud_rate divisor;
   -- GET(SELECTION): Setup the proper rate;

-- OUTPORT (P2, SYSTEM.BYTE(NUMBER)): Put LSB on 3f8 or 2f8;

-- OUTPORT (P3, 00): Put MSB on 3f9 or 2f9;

-- OUTPORT (P1, 03): Set parity, stop bit as (8--NONE--1);

-- OUTPORT (P4, 03): Set moden control register;

-- OUTPORT (P3, 00): Disable line control register;

-- RATES (P1, P2, P3, P4): Setup the transmission rate;

2. Operation 2: External Nav_Sensor operation setup

*. Port I/O: Set up the connecting ports to transmit sensor_data.

*. Value: Type of the measurement value(position data or measurement data).

*. Send_Rate: How often the data flow is (set up the baud rate).

. -- Note: operation 2 is operated externally by user

3. Operation 3: LOGGER--Log and Convert

*. Operations a logger to process GET_CHAR and logging

-- OPEN (FILE, OUT_FILE, NAME=>"POS.OUT"): Open buffer file to write;

-- GET_CHAR: Function to get and return the value;

-- PUT (A_CHAR): put data on the screen;

-- PUT (THE_FILE, A_CHAR): Put (while convert) data in the file;

-- PUT (FILE, A_CHAR): Put (while convert) data in to buffer file;

-- CLOSE (FILE): Close buffer file;

-- CLOSE (THE_FILE): Close the raw data file;

*. A function GET_CHAR: Get value from the assigned sensor in generic type (SYSEM.BYTE) then convert the value into any desire format in the operation.

-- INPORT (PORT1,LINE): Check 3fd or 2fd for data available;

-- TSTBIT (INTEGER(LINE), 0): Find out the status;

-- CLRBIT (LINE_INT, 0): Reset the test bit;

-- OUTPORT (PORT1, SYSTEM.BYTE(LINE_INT)): Assign value to 3fd or 2fd;

-- Return VALUE of CHARACTER'VAL(DATA);

4. Operation 4: DATA_LOGGER-- the main program unit

*. Logging the imported sensor_data

-- INITIAL (PORT1, PORT2): Initialization of the system communication;

-- BAUD_RATE: Setup the transmission rate;

-- OPEN (THE_FILE, IN_FILE, NAME=>"POS.DAT"): Open file to store data;
-- KEYPRESS (CURRENT_INPUT): Hot key control function;
-- POSITION_NOW: Display and print out the hard copy;
-- LOGGER (PORT1, PORT2): Log sensor data from the setup com_port;
-- CLOSE (THE_FILE): Close the raw data file;
-- EXCEPTION: Handle the exception commands;

5. Operation 5: POSITION_NOW--Display current data string
*. Operations a display of data string as user desired.
-- OPEN (FILE, IN_FILE, NAME=> "POS.OUT"): Open buffer file to read;
-- GET (FILE, ITEM): Get from buffer file;
-- PUT (ITEM): Put the data (ITEM) on the screen;
-- CLOSE (FILE): Ready for other operation;

-- Note: We characterized Data_Logger as a Transducer Task for the whole Suitcase Navigation System. By the definition of Booch's tasks, A transducer is a task that both calls entries of other tasks and also provides a service to other units (such as process & display status in Suitcase Navigation System, or Integration System in LCCDS).
-- Note: The first iteration of this project does not use tasking for program coding, but the following updates may want to use tasking, because of more concurrent operations

E. ESTABLISH THE COMMUNICATION

To establish the communication we must consider the relationship among objects, now that we have identified the operations of each object in our system, we may start from the external Navigation sensor. The operation of the external navigation sensor is completely isolated from any other object. But the sensor's data is coupled to other operations. We call the data SYSTEM. BYTE, and the function to get the data GET_CHAR:

GET_CHAR must be able to see SYSTEM.BYTE;

GET_CHAR is a function type of subprogram, it checks the line register of the COM_ PORT, and looking for the legal byte "[", which was assigned by the user and can be changed. The

return value of GET_CHAR is converted to CHARACTER, and distributed to different files and the display. A main subprogram that serves as the root of the system is required by Ada. We shall call this subprogram LOGGER, and it will contain the function (GET_CHAR). Additionally, this unit will be responsible for process & status display in the future. As a result:

LOGGER must see GET_CHAR and POSITION_NOW;

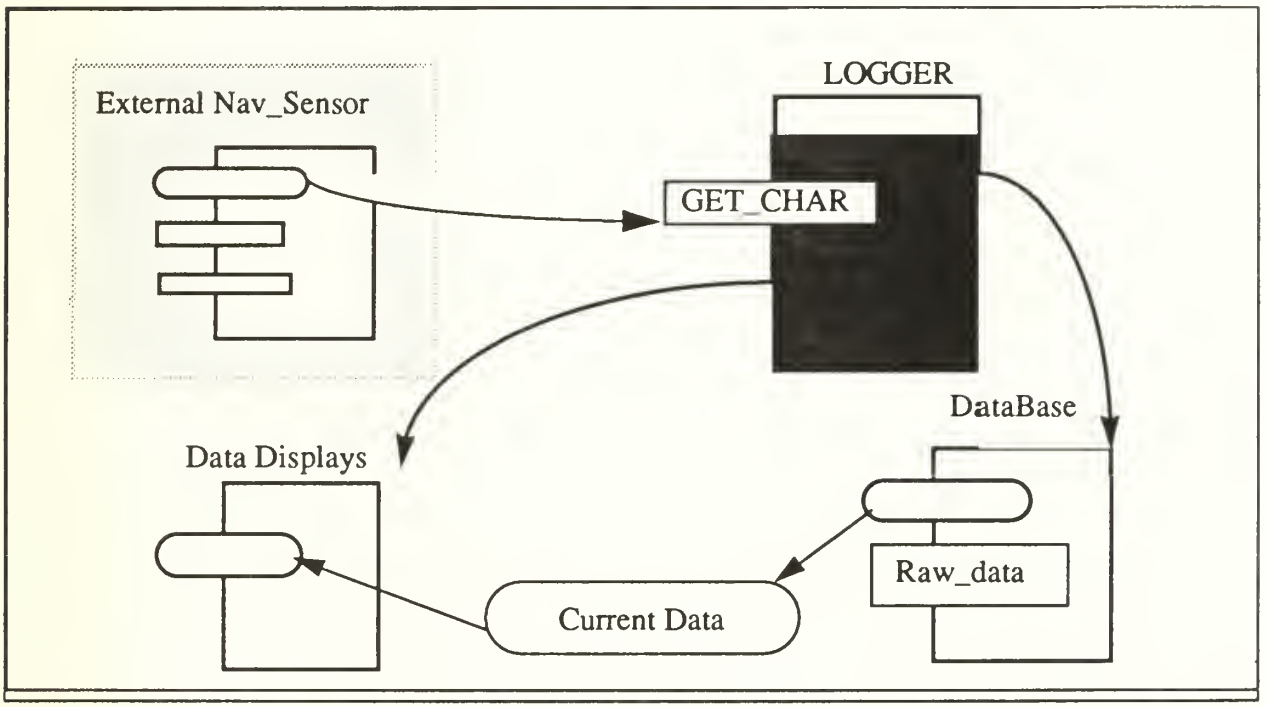

Figure-6: Description of Logger Design

Figure-6: illustrates these relationships. DATA_LOGGER is the main program, we will see four sub-units, which describes the whole operation:

-- INITIALIZATION: Subprogram process commands to handle user initialization.

-- BAUD_RATE: Subprogram process commands to establish proper communication.

-- POSITION_NOW: Subprogram to support the USER_COMMANDS.

-- LOGGER: Subprogram of periodic logging of sensor readings.

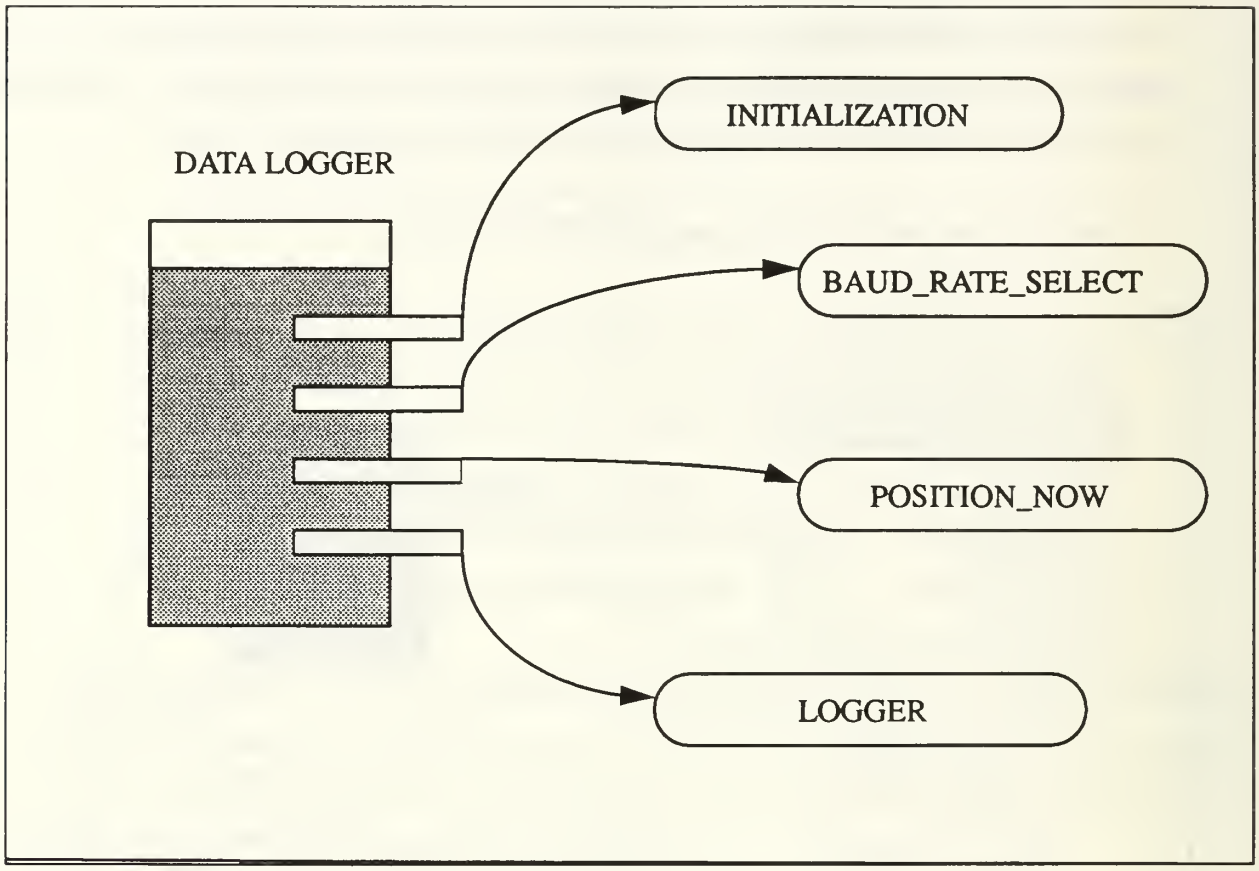

Figure-7: Description of Data_Logger Design

Figure-7: illustrates this level of the design. We also learned that at the interface of I/O ports the abstraction falls at a lower level in our design, and this ports are visible here only to the bodies of DATA_LOGGER. Figure-8: illustrates this level of the design.

F. ESTABLISH THE INTERFACE

The interface that we are concerned with here is mainly between Navigation sensor, Data_Logger, Printer, and User_Commands. In this thesis we are focused in the Navigation sensor interface through RS-232 and I/O ports interface from low_level to high_level language.

In early stage of this project, our lab has set up two PC connected by RS-232 as simulation system design for Suitcase Navigation System. Chapter V will present the simulation and

Appendix.1 introduces briefly the RS-232 interface for GPS (Trimble-4000). From the software designer's point of view, I/O ports interface (from low_level to high_level) to Ada is another main concern in this thesis. We have learned that Ada is capable to handle I/O in many different ways. In this project we use predefined package BIT (appendix.2), as port I/O interface support because package Bit can provide more control to the user. We will have more discussion in the next chapter about Data_Logger implementation.

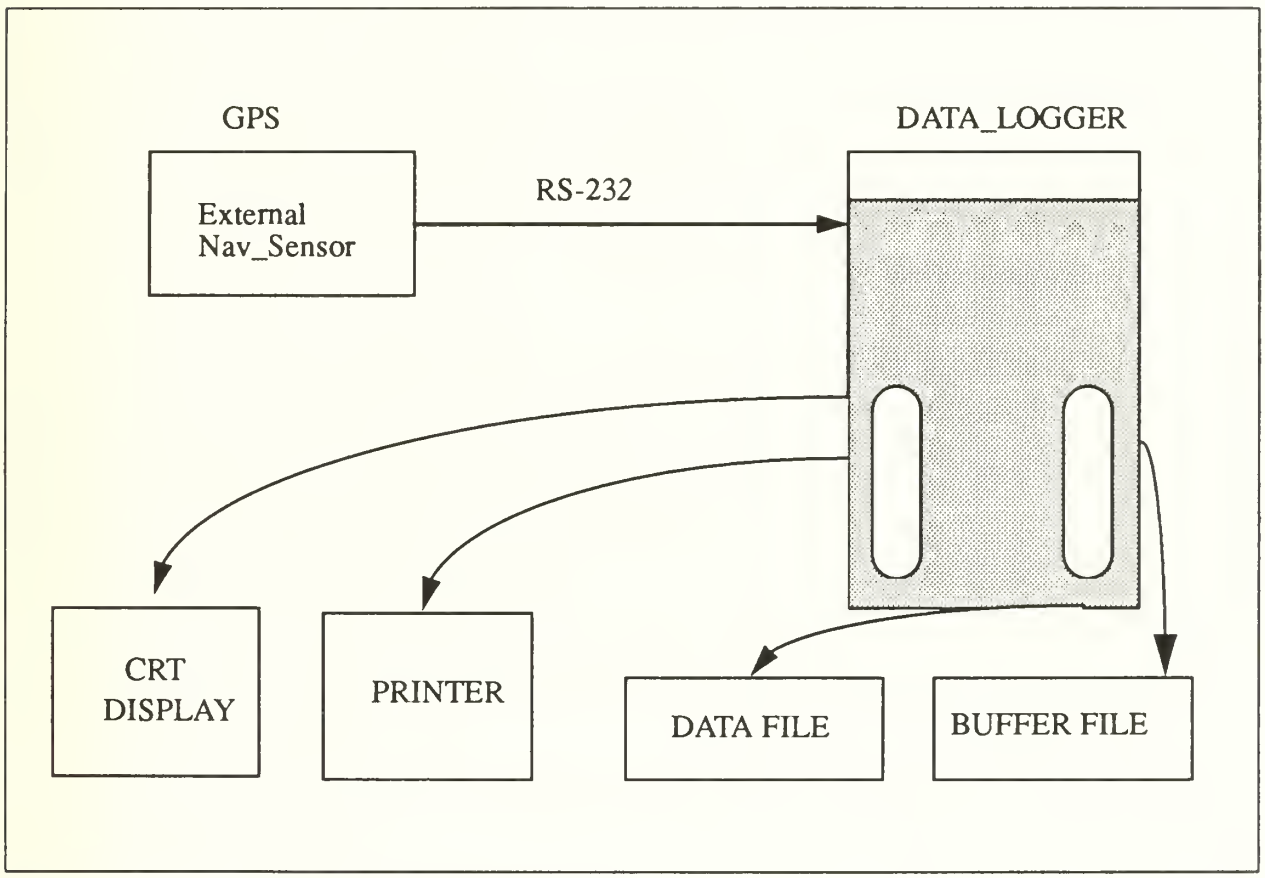

Figure-8: Description of Data_Logger I/O ports

IV. DATA_LOGGER IMPLEMENTATION

A. SYSTEM-SENSOR INTERFACING

To set up the interface between system and external sensor, we need to initialize the COM_PORT which is physically connected to the sensor. According to the pre-defined package BIT (Appendix.2), we have the procedures of SETBIT, CLRBIT which can set the bit number into the specified port and clear it when done, the function TSTBIT with in out VALUE in type INTEGER and assigned BIT number in type of BIT_NUM, which is subtype of INTEGER and range 0 .. 15. Also we have the capabilities to handle the VALUE in out as user assigned PORTNUM then put and get the VALUE in type SYSTEM.BYTE by using procedures INPORT and OUTPORT.

This is the design that we assign the COM_PORT:

Code-1: Initialization

---

* First of all we want to set up the communication through the standard RS-232 connections,

a procedure named INITIAL with two variables of type INTEGER, programmed to make the

selection of COM_PORT (3f8 or 2f8), and set the status register ready (3fd or 2fd).

* Needs to be done every time when restart the system.

```
: -- A procedure to start the program
procedure INITIAL(PORT1, PORT2: out INTEGER) is
    NUM: INTEGER;
  begin
      GET(NUM); -- to set up com_port number here
      case NUM is
          when 1 =>
              PORT1:= 1021;
              PORT2:= 1016;
              PUT_LINE("You've set Port1 = 1021, and Port2 = 1016");
          when 2 =>
```

```
            PORT1:= 765;
            PORT2:= 760;
            PUT_LINE("You've set Port1 = 765, and Port2 = 760");
          when others =>
              PUT_LINE("Wrong key! Sorry, you should be more careful ");
    end case; end INITIAL;
```

Code-2: Setup the Baud_Rate

---

* Secondly we setup the BAUD_RATE to complete the initialization of the system connectior and communication transmit through RS-232.

```
-- A procedure to set the system baud_rate
 procedure BAUD_RATE is
       CH: INTEGER;
     procedure RATES(P1, P2, P3, P4: in INTEGER) is
       NUMBER, SELECTION: INTEGER;
    begin
        OUTPORT (P1, 128); -- access Baud-Rate- Divisor
        GET(SELECTION);
        case SELECTION is
          when 1 =>
            NUMBER:= 96; -- Baud-Rate = 1200
          when 2 =>
            NUMBER:= 48; -- Baud-Rate = 2400
          when 3 =>
            NUMBER:= 24; -- Baud-Rate = 4800
          when 4 =>
            NUMBER:= 12; -- Baud-Rate = 9600
       end case;
       OUTPORT (P2, SYSTEM.BYTE(NUMBER)); -- convert the number into system.byte
```

```
-- put LSB on 3f8 or 2f8 (com1 or com2)set Baud
      OUTPORT (P3, 00); -- put MSB on 3f9 or 2f9
      OUTPORT (P1, 03); -- no p one s 8 bit
      OUTPORT (P4, 03); -- set modem control register
      OUTPORT (P3, 00); -- interrupt control register (disable)
  end RATES;
begin
    GET(CH);
    case CH is
      when 1 =>
         RATES(1019, 1016, 1017, 1020); -- call subprogram RATES and passed the var value
      when 2 =>
        RATES(763, 760, 761, 764);
    end case;
end BAUD_RATE;
```

External sensor operation requires certain levels of set up process operate by user. As a good example, Appendix.1 presents a brief introduction of Trimble 4000 GPS using RS-232 interface to an external computer. Operation requires three basic things; such as Port I/O match, type of VALUE declaration, and transmission rate set up. One way or the other, the practical cable connection and interfacing set up procedures should be carefully done step by step and may be done by using a reliable little test program.

B. SENSOR DATA HANDLER

Before we receive sensor data from an external sensor, the system shall be ready for data input. Appendix.3 provides the formats of the sensor data, which we are interested in using in this project. In Ada, we like to use file as a main memories unit to help user handle data in and out. To make room and assign an address to an incoming data_string, we could simply use:

```
OPEN (TYPE => FILE_TYPE, MODE => OUT_FILE, NAME =>: "POS.DAT");
```

This is to open a file as a text FILE_TYPE, with mode as OUT_FILE (WRITE) and named POS.DAT. The system is then ready for PUT or WRITE some data into the file. While data is coming from the COM_PORT, where it is going to rest, what is the format of data, and how can we handle it is becoming our next concern.

Since we decided to use predefined package BIT, we know we are getting data in BYTEs, and convert it into any type we need in the whole program. The main subprogram LOGGER that serves as the root of the system is programmed in this manner. It checks the line register for legal (defined by user) character, then log in the desired one. By the time when user monitor the input data on the screen, it stored into assigned file as well (Raw_data). On the other hand, our system keep the updated data_string in another buffer file called POS.OUT which store only the new updated position data, and ready to be used any time when the complete set of string is received.

Code-3: A main subprogram

---

* A main subprogram which contains the function GET_CHAR to get data from external navigation sensor and return CHARACTER data, then distributes data to data_file, buffer_ file, and display.

```
-- A procedure to logging data

procedure LOGGER (PORT1, PORT2: in INTEGER) is
-- port1 = 1021 | 765, port2 = 1016 | 760
      A_CHAR: CHARACTER;
      VALUE: SYSTEM.BYTE;
-- A function to get data from GPS sensor
      function GET_CHAR return CHARACTER is
         LINE, DATA: SYSTEM.BYTE;
         LINE_INT: INTEGER;
      begin
          loop
            INPORT (PORT1, LINE); -- check 3fd input data is available
               exit when TSTBIT(INTEGER(LINE), 0); -- loop until it's available
```

```
            if KEYPRESS then-- exit the loop
              raise ABORT_REQUEST;
            end if;
        end loop;
        LINE_INT:= INTEGER(LINE); -- convert LINE into integer
        CLRBIT(LINE_INT, 0); -- clear TESTBIT
        INPORT (PORT2, DATA); -- read data from 3f8
        OUTPORT(PORT1, SYSTEM.BYTE(LINE_INT)); -- put value back to 3fd
      return CHARACTER'VAL(DATA);-- return value in character
      end GET_CHAR;
begin
    loop-- wait for the first legal char
      OPEN (FILE, OUT_FILE, NAME => "POS.OUT"); -- open a temp out file
      while GET_CHAR /='[' loop-- process each message received
        null; -- and wait for start character
      end loop;
      PUT ('['); -- put start char on the screen
      PUT (THE_FILE,'['); -- put start char to file
      PUT (FILE,'['); -- put start character to temp file
      loop
        A_CHAR:= GET_CHAR; -- get and put the message
        PUT (A_CHAR); -- put data on the screen
        PUT (THE_FILE, A_CHAR); -- and to the file
        PUT (FILE, A_CHAR); -- and to the temp file
        exit when A_CHAR =']'; -- line 185
      end loop; -- get another set of data
      CLOSE(FILE); -- close temp file here
      exit when KEYPRESS; -- exit loop
    end loop;
end LOGGER;
```

C. SYSTEM DATA MANAGEMENT

The system is getting data from an external sensor and it is coming in like a flow. Procedure LOGGER has the control of coming data flow by using line register checking, and so we can keep the data as clear as we want. It is the problem that how can we keep and get a real_time data from the system. Procedure POSITION_NOW is presenting a simple way to solve this problem, by using a separate procedure we make the buffer file POS.OUT play an independent work while system running and it provides real_time data. This design is very useful for Suitcase Navigation System at the present time, when the user needs the position in a good accuracy. It is also good for future system implementation, which needs a whole set of independent position data_string.

Code-4: Display current position data

---

* To responds the user's need in between operation, the system can be interrupted by a single user command, to pull out the current position data and be able to print.

```
-- A procedure to print the desire data_string

procedure POSITION_NOW is
    ITEM: CHARACTER;
    FILE: FILE_TYPE;
begin
    OPEN (FILE, IN_FILE, NAME => "POS.OUT"); -- open the update file
    while not END_OF_FILE(FILE) loop
      GET(FILE, ITEM); -- get the data
      PUT(ITEM); -- put it on the screen
    end loop;
    CLOSE(FILE);
end POSITION_NOW;
```

D. PROGRAM CODING

We programmed the system DATA_LOGGER for PC or LapTop which is supported by DOS operating system. Program coding was done in IntegrAda and will be refined later, when the project move up to automation boundary, and multi-tasking. Since this is the first software program for Suitcase Navigation System, our main effort is to make the basic function work. The goal is clear, the structure of the whole system is firm, the program coding is as good as any other Ada program, which is as follows:

```
--------------------------------------------------------------------------
-- UNIT_NAME          | DATA_LOGGER.A
-- UNIT_DESCRIPTION   | This program is designed for Suitcase Navigation
--                    | data logger which logs data from GPS sensor store,
--                    | puts it into a file and prints the data string by the
--                    | user desire command
-- INPUTS             | GPS input raw data
-- OUTPUTS            | data record
-- CREATED            | 12 May 1991
-- AUTHOR             | CDR. Chin, Yu-Chi
-- ADVISOR            | DR. Uno R. Kodres
--------------------------------------------------------------------------
 with BIT, KEYBOARD, TEXT_IO, RTEXT_IO;
 use BIT, KEYBOARD, TEXT_IO, RTEXT_IO;

procedure DATA_LOGGER is

 -- program specification declaration to expose the parameters in out
  procedure INITIAL(PORT1, PORT2: out INTEGER);
  procedure LOGGER (PORT1, PORT2: in INTEGER);
  package INTEGER_INOUT is new INTEGER_IO(INTEGER);
  use INTEGER_INOUT;
 -- global variable declaration
  FILE, THE_FILE     : FILE_TYPE;
  PORT1, PORT2, ROCK : INTEGER;
  ABORT_REQUEST      : EXCEPTION;
```

```
-- A procedure to start the program
 procedure INITIAL( PORT1,PORT2  :  out INTEGER ) is
  NUM: INTEGER;
begin
  PUT_lINE("   AUTHOR   :  CDR. CHIN, YU-CHI R.O.C.N ");
  NEW_LINE;
  PUT_LINE("   ADVISOR  :  DR. UNO R. KODRES NPGS U.S.A. ");
  NEW_LINE;
  PUT_LINE("   DATE     :  16 MAY 1991 ");
  delay 5.0;
  NEW_LINE(24);
  PUT_LINE("A brief guide to use this program :");
  NEW_LINE;
  PUT_LINE("1. Select COM_PORT & Make sure your system Baud_Rate is match");
  NEW_LINE;
  PUT_LINE("2. Run the procedure DATA_LOGGER at proper COM_PORT");
  NEW_LINE;
  PUT_LINE("3. Use Space_Bar and then Prt_Sc to get updated data_string");
  NEW_LINE;
  PUT_LINE("4. Use (q) or (Q) to exit the program");
  NEW_LINE;
  delay 5.0;
  PUT_LINE("Give the COM_PORT number in (1) or (2)");
  GET(NUM);                          -- to set up com_port number here
    case NUM is
      when 1 =>
        PORT1  :=  1021;                    -- line register 3fd
        PORT2  :=  1016;                    -- com_port number 3f8
        PUT_LINE("You've set Port1 = 1021, and Port2 = 1016");
      when 2 =>
        PORT1  :=  765;                     -- line register 2fd
        PORT2  :=  760;                     -- com_port number 2f8
        PUT_LINE("You've set Port1 = 765, and Port2 = 760");
      when others =>
```

```
        PUT_LINE(" Wrong key !! Sorry, you should be more careful ");
      end case;
  end INITIAL;
-- A procedure to set the system baud_rate
  procedure BAUD_RATE is
    CH  :  INTEGER;

    procedure RATES(P1, P2, P3, P4 : in INTEGER) is
      NUMBER,
      SELECTION  :  INTEGER;
    begin
      OUTPORT (P1, 128);                    -- access Baud-Rate- Divisor
      PUT_LINE ("SET THE SYSTEM BAUD-RATE AS YOUR DESIRE");
      PUT_LINE (" --- SELECTION IS --- ");
      PUT_LINE ("    1  =  1200      ");
      PUT_LINE ("    2  =  2400      ");
      PUT_LINE ("    3  =  4800      ");
      PUT_LINE ("    4  =  9600      ");
      NEW_LINE;
      GET(SELECTION);
       case SELECTION is
         when 1 =>
           NUMBER := 96;                        -- Baud-Rate = 1200
           PUT_LINE ("YOUR SYSTEM BAUD-RATE IS SET TO 1200 NOW");
         when 2 =>
           NUMBER := 48;                        -- Baud-Rate = 2400
           PUT_LINE ("YOUR SYSTEM BAUD-RATE IS SET TO 2400 NOW");
         when 3 =>
           NUMBER := 24;                        -- Baud-Rate = 4800
           PUT_LINE ("YOUR SYSTEM BAUD-RATE IS SET TO 4800 NOW");
         when 4 =>
           NUMBER := 12;                        -- Baud-Rate = 9600
           PUT_LINE ("YOUR SYSTEM BAUD-RATE IS SET TO 9600 NOW");
         when others =>
```

```
        PUT_LINE (" WRONG SELECTION !! TRY AGAIN !! ");
      end case;
    OUTPORT (P2,SYSTEM.BYTE(NUMBER)); -- convert the number into system.byte
                    -- put LSB on 3f8 or 2f8 (com1 or com2)set Baud
    OUTPORT (P3, 00);       -- put MSB on 3f9 or 2f9
    OUTPORT (P1, 03);       -- 8 bit, no parity, one stop ( 8--NONE--1 )
    OUTPORT (P4, 03);       -- set modem control register
    OUTPORT (P3, 00);       -- interrupt control register (disable)
    NEW_LINE;
    PUT_LINE ("BAUD-RATE SETTING IS DONE !! ");
  end RATES;

begin
  NEW_LINE(5);
  PUT_LINE("Now The program is design to setup the system BAUD_RATE");
  NEW_LINE;
  PUT_LINE("Make your choose for COM1 or COM2 : use  -1-  or  -2-");
  GET(CH);
    case CH is                          -- set the value to ports
      when 1 =>
        RATES(1019, 1016, 1017, 1020);          -- set baud_rate and parity
        PUT_LINE("FOR COM1 SETTING");           -- to com_port # 1
      when 2 =>
        RATES(763, 760, 761, 764);              -- set baud_rate and parity
        PUT_LINE("FOR COM2 SETTING");           -- to com_port # 2
      when others =>                    -- line 142
        PUT_LINE("WRONG KEY !! TRY AGAIN !!!");
    end case;

end BAUD_RATE;

-- A procedure to print the desire data_string
procedure POSITION_NOW is
  ITEM  :  CHARACTER;
```

```
  FILE  :  FILE_TYPE;

begin
  NEW_PAGE;
  NEW_LINE(5);
  PUT_LINE(" The current data_string is : ");
  NEW_LINE;
  OPEN (FILE, IN_FILE, NAME => "POS.OUT" );  -- open the update buffer file
  while not END_OF_FILE(FILE) loop          -- loop to get the whole string
    GET(FILE, ITEM);                        -- get the current data_string
    PUT(ITEM);                              -- put it on the screen
  end loop;                                 -- out loop when done
  NEW_LINE(10);
  CLOSE(FILE);                              -- close file back to survey
end POSITION_NOW;

-- A procedure to log data from external nav_sensor
procedure LOGGER (PORT1, PORT2  :  in INTEGER) is
                                -- port1 = 1021l765, port2 = 1016l760
  A_CHAR       : CHARACTER;
  VALUE        : SYSTEM.BYTE;

  -- A function to get data from GPS sensor
  function GET_CHAR return CHARACTER is
    LINE, DATA : SYSTEM.BYTE;
    LINE_INT   : INTEGER;

  begin
    loop
      INPORT (PORT1, LINE);             -- check 3fd or 2fd is available
      exit when TSTBIT(INTEGER(LINE), 0);  -- loop until it's available
    end loop;                           -- line 182
    LINE_INT := INTEGER(LINE);          -- convert LINE into integer
    CLRBIT(LINE_INT, 0);                -- clear TESTBIT
```

```
    INPORT (PORT2, DATA);              -- read data from 3f8 or 2f8
    OUTPORT(PORT1, SYSTEM.BYTE(LINE_INT)); -- put value back to 3fd or 2fd
    if INTEGER(DATA) > 127 then        -- a test loop for bad data
      DATA := 42;                      -- if not a character use '*'
    end if;
    return CHARACTER'VAL(DATA);
  end GET_CHAR;

begin
  PUT_LINE("Waiting for the first legal character");
  loop                               -- wait for the first legal char
    OPEN (FILE, OUT_FILE, NAME => "POS.OUT"); -- open a temp out file
    while GET_CHAR /= '[' loop       -- process each message received
      null;                          -- and wait for start character
    end loop;
    PUT ('[');                       -- put start char on the screen
    PUT (THE_FILE, '[');             -- put start char to file
    PUT (FILE, '[');                 -- put start cgar to temp file
    loop                             --
      A_CHAR := GET_CHAR;            -- get and put the message
      PUT (A_CHAR);                  -- put data on the screen
      PUT (THE_FILE, A_CHAR);        -- and to the file
      PUT (FILE, A_CHAR);            -- and to the temp file
      exit when A_CHAR = ']';
    end loop;                        -- get another set of data
    CLOSE(FILE);                     -- close temp file here
    exit when KEYPRESS;              -- exit loop
  end loop;
end LOGGER;
                                     -- main program starts here
begin
  INITIAL (PORT1, PORT2);            -- a brief program instruction
  BAUD_RATE;                         -- program to set system Baud_Rate
  OPEN (THE_FILE, OUT_FILE, NAME => "POS.DAT");-- make file pos.dat availabe
```

```
  loop
   if KEYPRESS(CURRENT_INPUT) then        -- press Space_Bar to interrupt and
    NEW_LINE;                             -- ready to print the screen
    PUT_LINE("New data_string is as follow:");
    NEW_LINE;
    ROCK:= KEY_VALUE;                     -- assign keypress to variable
    if ROCK = 32 then                     -- if Space_Bar is true then
     POSITION_NOW;                        -- bring the data to the screen
    elsif ROCK = 81 or ROCK = 113 then    -- press (q) to exit the program
     raise ABORT_REQUEST;                 -- raise exception
    end if;                               -- else program will keep running
   end if;                                -- no interruption
   LOGGER(PORT1, PORT2);                  -- program to log sensor data
  end loop;
  exception
   when ABORT_REQUEST =>                  -- exit the program
  CLOSE (THE_FILE);
  PUT_LINE ("END OF PROGRAM DATA_LOGGER.A");
end DATA_LOGGER;
--------------------------------------------------------------------------------
```

V. DESCRIPTION OF THE DATA_LOGGER

A. THE REQUIRED PROGRAM STRUCTURE

1. Hardware Structure

The Navigation Data_Logger System is structured by 4 parts, which includes Antenna, GPS, Computer (PC or LapTop), and Printer. Since the external navigation sensor will be a generic positioning device, we will use the name GPS as generic Global Positioning System. For this program, we use Trimble-4000S GPS as the target Nav_Sensors, because it is available at school. Before we have the real GPS, our lab has set up two 286-Zenith PC connected by RS-232 cable to simulate the communications. By transmit data from computer A and received by computer B, we have learned that the wire connection through RS-232 will cause no problems for the project. By the time we got the GPS (Trimble-4000), the system is set up similar as Figure-9, and the software was written to be tested.

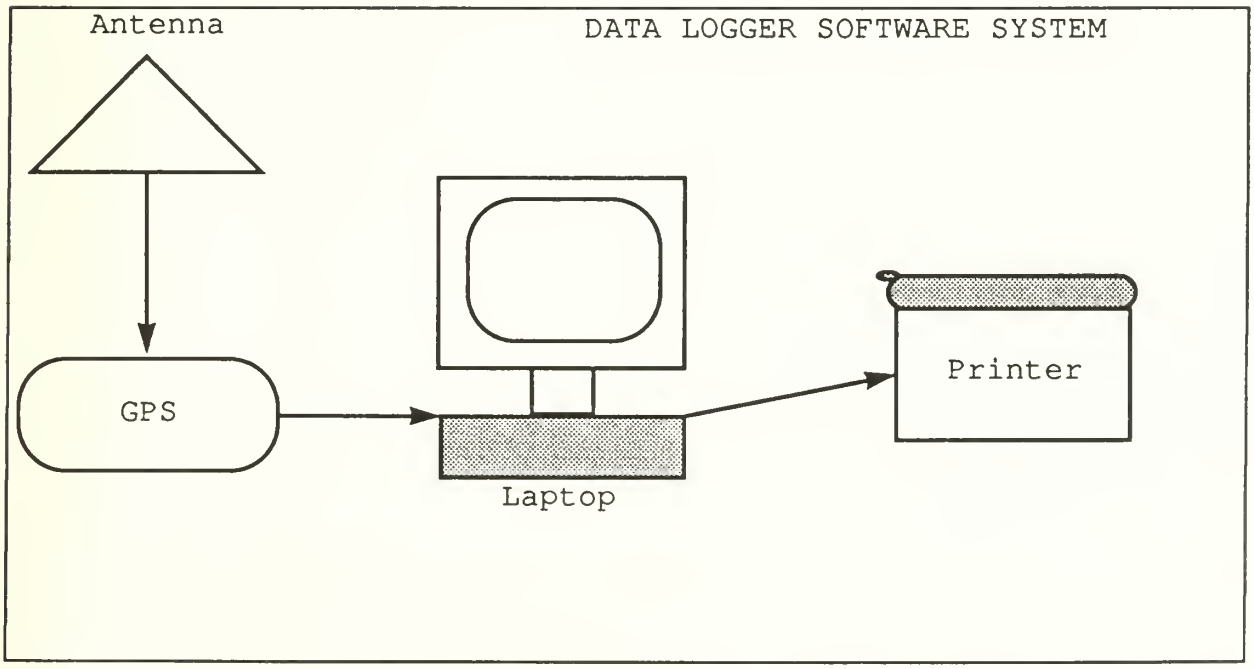

Figure-9: Hardware Structure of Data Logger

2. Software Structure

Ada is the only choice of software structure. As we have mentioned, Ada is well designed for the applications via concurrently, real-time control, exception handling, and unique input and output. But as well as the program needs, Ada provides low_level programming also. Because of the hardware constrains for the type of computer will be a LapTop or a 286 PC, the software is constrained to DOS supported program by definition. We went through the applications of Ada input/output and learned that programming for input/output (I/O) has always been like going to the dentist for a toothache, it's some-thing that language designers have to do, but they tend to put it off in the hope that it might go away.

Ada does not treat I/O as a thorn in the side of the language. Infact, with the extensibility provided by Ada's packaging mechanism and generic facilities, Ada does not have to provide any special language features to accommodate I/O. The user can build his own I/O routines for communication with unique devices. Furthermore, without adding any new language constructs, the user can use any predefined units for I/O of common data types, such as characters, integers, and real numbers, which can be selected as needed.

Here, we use TEXT_IO to handle the generic text_file I/O, which can support the data storage and transfer at the same time while the input data is presented on the screen it outputs to an existing file and a designed buffer file which stored the updated data_string and can be pull out by User_commands. We also use another predefined package BIT, to handle the port I/O. This package supports the capabilities of checking the status of assigned port, and be able to put and get I/O values from the ports, which gives the user a little bit more control over the external communication devices.

B. ADVANTAGES OF THE DATA_LOGGER

The Navigation Data_Logger System was the first Ada program of the Suitcase Navigation System. It satisfied the needs of SNS project requirement to log the positioning data in the logger, and also provides the screen display for easy reading, but most importantly it provides a single set of current position data_string, which is the real-time position that the operation

required. Any time when you need to mark the position or you need a fix on the open sea, the Space_Bar is the user command to bring the current position data_string to the screen. To get a hardcopy by using print screen won't disturb the program processing the Nav_Sensors, but provides the scientists an clear data_record, and a reliable global position.

C. USER's MANUAL

The instruction of how to use the Navigation Data_Logger System has been written as an operation process and responses in Appendix.5. Beyond that, user should be able to set up the external Nav_Sensor's function properly and capable to make the connection to the control system (computer) through RS-232. When the operation is on, user should know the basic serial COM_ PORT's function and how to adjust the transmission rate (Baud_Rate). After the hardware set up, the user should carefully start the initialization and run the program under the brief guide.

Since this program is directed to Trimble-4000s GPS, the logical position data_string was bounded by the special characters '[' and']'. It was programmed inside the procedure LOGGER and was very easy to redefined by change the boundary character and it's decimal representations. To avoid the programming language constrains, user can select the separated function program listed in Appendix.4 and collected the data as well.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The Suitcase Navigation System is designed for the people who do not know much about Navigation, but want to work in the environment which needs to collect this information. The project uses GPS as a fundamental sensor, to provide the crucial data. This thesis is an effort in developing a Navigation Data_Logger software system, that can satisfy the needs to process navigation sensor data.

In order to achieve this basic requirement, we start from data format analysis and single byte transmission, finally we programmed the Navigation Data_Logger software system. As we may recall from chapter-4, the performance of the Data_Logger is demonstrates that it is quite efficient, mainly for logging messages from GPS and transmitting data to Screen, Raw_data file, and Buffer file. User can view the data on the screen, store the data in Raw_data file, and get the hard copy from buffer file. The conclusion arise from our Navigation Data_Logger System design is that Ada is an high level language, but it can be used to handle low_level I/O, and support the software design in high level programming.

B. RECOMMENDED FOLLOW-ON WORK

As stated in Chapter I, this thesis is directed to the GPS , represented by Trimble-4000 for the Suitcase Navigation System. It is our hope that it can serve as a firm foundation for future and more enhanced implementations. During the process this thesis was being developed, many new ideas were brought up. We would like to give some suggestions for future enhancements in the system.

1) The database of Data_Logger should be the first concern of the program in the near future. For long term survey, we need a database manager support our Nav_Sensors data processor.

2) Different external sensors will be the second problem. The system I/O ports concurrence and real time programming can be support by Ada, a separate package abstraction of these I/O ports could be a high-level view of physical memory locations.

3) Hot_Key control, Ada exception handling and low-level features of I/O representation specification is highly recommended for concurrent real- time processing. As noted in Appendix.6 from Reference-3.

4) Remote control to external navigation sensor is a possible and the software to challenge the next Ada programmer. The Trimble manual and Appendix.1 explains further how this should be done.

APPENDIX--I GPS Trimble-4000S

Using the RS-232 interface for Trimble 4000.

1. Select proper cables and connectors to connected the GPS and computer.
--Select one of the connectors (ports) and make a connection in the way of Matching Pins.*
-- Note: * In general, the only signals that need to be connected are:

Trimble 4000	Computer Port
Ground	Ground
TXD (Transmit Data)	RXD (Receive Data)
RXD (Receive Data)	TXD (Transmit Data)

2. Turn on the system and wait until its self calibration completed.
-- Reference position needs to load at this moment. when CALIBRATION OK, and then
-- POSITION FIX 3,4, A display shows the GPS is in normal function.
3. Use the I/O key on the control panel to select I/O functions.
4. Select SETUP, BAUD, and the PORT you wish to use.
5. Set the Data_Logger receving rate match the GPS send rate.
--All Trimble Model 4000 GPS receivers defaults to 9600 Baud, 8-bits, odd parity, and one
--stop bit (8-ODD-1), Data_Logger set up 8-bits none parity, and one stop bit (8-NONE-1) set your GPS to match.
6. Select the I/O, and PRINT function to get the desire position data.
7. Remote control programming : To interact properly with the specific interface function desired.*
--Note: * The remote control interface allows an external computer to set the controls on
--GPS receiver and transfer the measurements and calculations to the computer.
8. Interact with a remote computer through a report session. *
--Note: * A description of The Report Session:

The Report Session is enabled by sending a special character, ATTENTION (35 D or 23 H). The ATTENTION command switches the port on which it is received to become the remote con trol port, and the next byte received will be logged as a command and will trigger a report session when the system cycle reaches that step. At that time, the command will be echoed. --

-- Report Session loop:

1. Wait for first byte of next command.
2. Echo the command.
3. Read any other bytes of command (no echoing)
4. Execute the command. This may include sending multiple bytes to the computer.

* Note : There can be a significant delay between sending the first command and the echo. This is especially true if the receiver is making synchronized measurements.

-- Something you have to know about Trimble-4000 remote control commands.

1. All commands start with a single byte that is evenly divisible by three.
2. The only other valid bytes are :
   35 d : ATTENTION;
   34 d : old-style ATTENTION;
   37 d : ACTIVITY REQUEST;
3. Any illegal bytes will cause an abort of the report session.
4. GPS will wait only 30 seconds for a command it will beep and exit the current session if more time than this elapses.
5. Command Formats:
   - [B] Bytes is single eight-bit INTEGER, range O .. 255;
     (e.g. 255 for ON and O for OFF)
   - [D] BCD " Binary-Coded-Decimal " digit is single byte range O .. 9;
     (e.g. Lat, Long. Hgt in command 51 d are sent as sequences of BCD digits.)
   - [R] Floating Point Real number. The real # represented is:
     [R] = (-1)s x 2 (exp-1023) x 1.mmmm.. mmmm.
     (i.e. IEEE 754 standard for long real numbers)
6. Unit of semicircles in use:
   1 semicircle = 180 degrees = Pi radians.

The Lat and Long values output over the remote control port are both sent n semicircles. The values are sent as 8-byte real numbers using IEEE standard format After converting to the receiving computer, the value will be in semicircles, range -1.0. +1.0 which means -180.0. +180.0 degrees.'+' = North & East,'-' = South & West. Another converting is desirable to include degrees, minutes, and seconds.

An example of semicircle to degrees :

(e.g. Semicircle = -0.6875833333

Degrees = 0.6875833333 * 180.0

= 123.765000 degrees

Minutes = fraction (123.765) * 60

= 0.765 * 60 = 45,9000 minutes

Seconds = fraction (45.9) * 60

= 0.900 * 60 = 54.00 seconds

Thus, (Longitude)

- 0.6875833333 (Semicircles)

= West 123.765000000 deg (decimal degrees)

or = West 123 deg 45.9000 min (degrees an decimal minutes)

or = West 123 deg 45 min 54.00 sec (degrees, minutes, seconds)

APPENDIX--II Predefined PACKAGE BIT

```
with SYSTEM;
use  SYSTEM;

 package BIT is

    subtype   BIT_NUM is INTEGER range 0 .. 15;
     procedure SETBIT(VAL : in out INTEGER; BIT : in BIT_NUM);
     procedure CLRBIT(VAL : in out INTEGER; BIT : in BIT_NUM);
     function  TSTBIT(VAL : in INTEGER; BIT : in BIT_NUM) return BOOLEAN;
     function  LAND(VAL1, VAL2 : in INTEGER) return INTEGER;
     function  LOR(VAL1, VAL2  : in INTEGER) return INTEGER;
     function  LXOR(VAL1, VAL2 : in INTEGER) return INTEGER;
     function  LNOT(VAL : in INTEGER) return INTEGER;
     function  PEEK (ADDR : in INTEGER) return BYTE;
     procedure POKE (ADDR : in INTEGER; VAL : in BYTE);
     function  PEEK (SEGMNT, OFFSET : in INTEGER) return BYTE;
     procedure POKE (SEGMNT, OFFSET : in INTEGER; VAL : in BYTE);
     function  CODE_SEG return INTEGER;
     function  DATA_SEG return INTEGER;
     procedure INPORT (PORTNUM : in INTEGER; VALUE : out BYTE);
     procedure OUTPORT (PORTNUM : in INTEGER; VALUE : in BYTE);

end BIT;
```

APPENDIX--III SAMPLE DATA

Different set of data from GPS and RPS

```
TRIMBLE 4000 GPS RECEIVER SETUP PARAMETERS  MON 098 08-APR-91
21:24:07
-----------------------------------------

GPS WEEK NUMBER:        0587
REFERENCE LATITUDE:     36:35.9000N
REFERENCE LONGITUDE:    121:52.5990W
REFERENCE HEIGHT:       -0020.0 meters
FIXED HEIGHT:           YES
REFERENCE FREQUENCY:    +0.000E+00 delta f/f
FIXED FREQ:             YES
LATITUDE  OFFSET:       00:00.0000N
LONGITUDE OFFSET:       000:00.0000E
HEIGHT    OFFSET:       +0000.0 meters

MASKS
-----
PDOP MASK:  07.0
ELEVATION MASK:  10 degrees
DATA AGE SINCE DECODE MASK:  03 hours

CALIBRATION DATA
----------------
CODE CALIBRATION:  +00.01 +00.01 -00.06 -00.04 +00.06 -00.67 -01.77
-00.08 +00.
02 meters
CARRIER PHASE CALIBRATION:  +000 +000 +000 +000 +000 +000 +000 +000
+000 mm
CALIBRATION TIME:  0010 secs

SV STATUS
---------
HEALTHY SVS:  02 03 06 11 12 13 14 15  &
              16 17 18 19 20 21 23
DISABLED SVS:
IGNORE HEALTH SVS (POSITIONING):
IGNORE HEALTH SVS (SURVEY):
AUTO SV SELECTION:  MIN SVS 3,  MAX SVS 4
AUTO INITIAL SV SEARCH
ALL-IN-VIEW SOLUTION:  ENABLED
```

```
RS232 PORTS
-----------
PORT1:   9600 BAUD 8-ODD--1  START CC= 000 000 000   STOP CC= 000
000 000
        POSITION, MEASUREMENTS
PORT2:   9600 BAUD 8-ODD--1  START CC= 000 000 000   STOP CC= 000
000 000
        OFF
POSITION PRINT TIMER:  00 mins
PRINT ID: 00

DATA BASE
---------
ION  UPDATE:  ENABLED
BUL  UPDATE:  ENABLED
DATA UPDATE:  ENABLED

RECEIVER OPERATION
------------------
DOPPLER AIDING:  ON
SYNC:            002.0 secs
OSCILLATOR DAC:  ON
SV AVERAGING:    01.0 secs
POSITION AVERAGING:  0001

SOFTWARE
--------
NAVIGATION PROCESSOR SOFTWARE VERSION:  3.24  16-AUG-89
SIGNAL PROCESSOR SOFTWARE VERSION:      3.20  23-MAR-89
```

```
PART ONE
--------
FROM TRIMBLE 4000S
FUNCTION PRINT MODE POS (position):

 WN    REF.LAT     REF.LON    REF.HGT LAT.OFFSET   LON.OFFSET
HGT.OFF SV,POSAV
G PDOP EL AGE MIN MAX  REF.FRQ
0587 36:35.9000N 121:52.5990W -0020.0 00:00.0000N 000:00.0000E
+0000.0 01.0 000
1 07.0 10  3   4 +0.000E+00
 ID DAY DOY   DATE      TIME    LATITUDE    LONGITUDE    HGT  PDOP
CLOCK   V.VE
L  H.VEL   HDG  FREQ.OFFSET CONT S  SVS
[00 MON 098 08-APR-91 21:24:03 36:35.9482N 121:52.5926W -0017 03.8
309263 +000.
03 000.00 000.0 +5.4522E-11 0043 4 18,19,11,2]
[00 WED 059 28-FEB-90 07:08:13 36:18.2773N 121:59.7371W -0035 04.3
687719 +000.01 002.76 173.8 +4.2469E-11 0024 4 11,3,12,19]
[00 WED 059 28-FEB-90 07:08:29 36:18.2653N 121:59.7351W -0035 04.3
687720 -000.00 002.79 169.6 +4.4440E-11 0025 4 11,3,12,19]
[00 WED 059 28-FEB-90 07:08:44 36:18.2569N 121:59.7421W -0020 03.2
687785 +000.00 002.78 167.0 +3.2741E-10 0026 3 3,12,19]
[00 WED 059 28-FEB-90 07:08:59 36:18.2455N 121:59.7375W -0020 03.2
687785 +000.00 002.84 160.5 +9.1577E-11 0027 3 3,12,19]
[00 WED 059 28-FEB-90 07:09:32 36:18.2230N 121:59.7229W -0020 03.2
687791 +000.00 002.85 143.9 +3.8273E-11 0001 3 3,12,19]
[00 WED 059 28-FEB-90 07:09:50 36:18.2134N 121:59.7046W -0026 02.2
687757 -000.01 002.75 133.4 +7.8132E-11 0001 4 11,3,12,6,19]
[00 WED 059 28-FEB-90 07:10:05 36:18.2060N 121:59.6937W -0026 02.2
687758 +000.00 002.64 125.3 +7.4215E-11 0002 4 11,3,12,6,19]
[00 WED 059 28-FEB-90 07:10:20 36:18.2004N 121:59.6820W -0026 02.2
687760 +000.02 002.54 116.6 +7.5696E-11 0003 4 11,3,12,6,19]
[00 WED 059 28-FEB-90 07:10:35 36:18.1965N 121:59.6698W -0027 02.2
687761 +000.00 002.45 107.9 +7.5028E-11 0004 4 11,3,12,6,19]
[00 WED 059 28-FEB-90 07:10:50 36:18.1959N 121:59.6600W -0034 03.5
687746 -000.01 002.36 100.2 +7.4971E-11 0005 4 11,3,12,6]
SV EL AZM SN IODC CONT  GPS.TIME    AVG.LAT    AVG.LON    AVG.HGT
NO. RMSLAT LON   HGT  AVGCLK   CLK.RATE   AVG.FRQ
11 31 037 13 0462 0005 +285056.512  36:18.1959N 121:59.6600W -
0034.6 0001 00.00 00.00 00.00 687746 +0.0000E+00 +7.4971E-11 A
03 36 255 16 0391 0139 +285056.512
12 72 278 18 0029 0052 +285056.512
06 09 159 11 0228 0005 +285056.512
```

```
PART TWO
--------
FROM TRIMBLE 4000 XS
FUNCTION PRINT MODE MEAS:


[ +1.62465152000E+08 +2.97336525754E+05 +4.09017287885E+06 -
5.79796142489E+05 -
5.79951245869E+05 +5.19603100000E+06 +2.08923525974E+07
+2.97319082170E+05 +1.8
0110329726E-01 +0.00000000000E+00 +0.00000000000E+0000900102218][
+1.6246515200
0E+08 +3.470368450
75E+04 +4.09173391062E+06 +4.02280631719E+05 +4.02282417503E+05
+5.18200400000E
+06 +2.06316314375E+07 +3.46966031546E+04 +7.29198695187E-02
+0.00000000000E+00
 +0.00000000000E+0000300102019][ +1.62465152000E+08
+5.47442764476E+04 +4.09390
777905E+06 +6.6662
4958540E+04 +6.66629219876E+04 +1.79614000000E+05
+2.35639205651E+07 +5.4726886
2247E+04 +4.14212197467E-02 +0.00000000000E+00
+0.00000000000E+0000200000811](
+2.03328483276E-01 -6.77091561631E-01 -6.56408896315E+00 -
2.70732416499E+06 -4.
35348044922E+06 +3
.78176743118E+06 +2.83064040661E+00 +9.27233423855E+04 -
7.74005685580E-03)[00 M
ON 098 08-APR-91 21:07:53 36:35.9477N 121:52.5885W -0006 02.8
309292 +000.00 00
0.02 267.7 +2.2601E-11 0014 3 18,19,11]
[ +1.62480256000E+08 +2.80252655781E+03 +4.09016570002E+06 -
5.85057548887E+05 -
5.85212616397E+05 +5.21113400000E+06 +2.08976139552E+07
+2.78805384002E+03 +1.1
0563354440E-01 +0.00000000000E+00 +0.00000000000E+0000500102318][
+1.6248025600
0E+08 +3.547620969
67E+04 +4.09173033829E+06 +4.01510749528E+05 +4.01512559628E+05
+5.19710800000E
+06 +2.06324012107E+07 +3.54664817960E+04 +2.31367014084E-02
+0.00000000000E+00
 +0.00000000000E+0000100101819][ +1.62480256000E+08
+4.92651948358E+04 +4.09390
054617E+06 +7.2135
```

PART THREE

FROM MINI-RANGER FALCON 484
FUNCTION PRINT POSITION :

```
 05:48:26.4 15  29739.9 1  6     1.0 13   3403.8 1  6     0.9  1
17718.9 1 19
    2.8  4  23691.4 1 38    -1.4    608236.3   4073620.7   0.1 F4
0 C    3.6
 2.9\TTAG:  380343\   9\
  05:48:28.4 15  29738.9 1  6     1.4 13   3402.5 1  6     1.7  1
17718.9 1 19
     3.4  4  23691.6 1 37    -1.1    608237.3   4073621.1   0.2 F4
0 C    3.6
  3.6\TTAG:  380367\   0\
  05:48:30.4 15  29737.9 1  6     1.9 13   3400.3 1  6     3.3  1
17718.6 1 19
     4.3  4  23691.4 1 38    -0.2    608238.0   4073621.5   0.3 F4
0 C    3.6
  4.9\TTAG:  380410\   0\
  05:48:32.4 15  29738.7 1  6     1.4 13   3402.5 1  6     1.8  1
17718.6 1 19
     3.6  4  23691.5 1 38    -1.1    608237.7   4073621.8   0.1 F4
0 C    3.6
  3.8\TTAG:  380433\   0\
  05:48:34.4 15  29738.3 1  6     1.5 13   3403.0 1  6     1.6  1
17718.4 1 19
     3.7  4  23691.7 1 38    -1.5    608237.4   4073621.4   0.0 F4
0 C    3.6
  3.9\TTAG:  380457\   0\
  05:48:36.4 15  29739.0 1  6     1.7 13   3401.3 1  6     2.5  1
17718.5 1 19
     3.9  4  23691.4 1 38    -0.7    608237.6   4073621.2   0.1 F4
0 C    3.5
  4.2\TTAG:  380499\   0\
  05:48:38.4 15  29739.5 1  6     1.6 13   3401.9 1  6     2.2  1
17718.0 1 18
     4.0  4  23691.5 1 39    -1.1    608237.9   4073621.6   0.1 F4
0 C    3.6
  4.2\TTAG:  380523\   0\
  05:48:40.4 15  29739.4 1  6     1.7 13   3400.8 1  6     2.7  1
17718.6 1 18
     3.9  4  23691.6 1 37    -0.5    608238.6   4073621.3   0.2 F4
0 C    3.6
  4.2\TTAG:  380546\   0\
```

APPENDIX--IV Ada Program Examples

```
 --Example 1
 ------------------------------------------------------------------------------
-- UNIT_NAME          | DATA_IN.A
-- CSCI_NAME
-- UNIT_DESCRIPTION    | THIS PROGRAM IS DESIGNED FOR RECIEVED DATA FROM
--                  EXTERNAL DEVICE AND DISPLAYED ON THE SCREEN.
-- INPUTS             | GPS DATA FILES i.e. line56.dat
-- OUTPUTS            | read on the screen
-- CREATED            | 13 March 1991
-- AUTHOR             | Chin, Yu-chi
-- ADVISOR            | Dr. Uno R. Kodres
-- DATE ------------ AUTHOR -------- REVISION # -- PR # -----TITLE ------------
-- 0313/1991        Chin, Yu-chi       1                DATA_IN.A
------------------------------------------------------------------------------
with BIT, TEXT2_IO;
use BIT, TEXT2_IO;

procedure DATA_IN is
 PORTNUM  :  INTEGER  := 1016;
 VALUE    :  SYSTEM.BYTE;
begin
 INPORT (PORTNUM,VALUE);
 PUT_LINE ( "Port number " &INTEGER'IMAGE(PORTNUM)&
        " is currently " &SYSTEM.BYTE'IMAGE(VALUE) );
 PUT_LINE ( "---------- DATA RECEIVED ----------");
end DATA_IN;
```

```
-- Example 2
 ------------------------------------------------------------------------------
-- DATE ------------ AUTHOR -------- REVISION # -- PR # -----TITLE ------------
 --9 April 1991   Chin, Yu-Chi          5                test12.a
------------------------------------------------------------------------------
 with BIT, TEXT_IO;
use BIT, TEXT_IO;
 procedure TEST12 is
 LINE_INT,
 PORTNUM   :   INTEGER ;
 LINE,
 DATA,
 VALUE    :   SYSTEM.BYTE;
 COUNT    :   INTEGER  := 0;
 package INTEGER_INOUT is new INTEGER_IO(INTEGER);
 use INTEGER_INOUT;
begin
 loop
  loop
   INPORT (1021, LINE);       -- check 3fd see if input data is available
   exit when TSTBIT(INTEGER(LINE),0);  -- loop until it's availab
  end loop;
  LINE_INT := INTEGER(LINE);    -- convert LINE into integer
  CLRBIT(LINE_INT,0);          -- clear 3fd
  OUTPORT(1021,SYSTEM.BYTE(LINE_INT));  -- put value back to 3fd
  INPORT (1016, DATA);         -- read data from 3f8
  PUT (CHARACTER'VAL(DATA));    -- put data on the screen
  exit when INTEGER(DATA) = 93; -- exit when input data is ']'
 end loop;
 delay 1.0;                    -- test delay
 PUT_LINE ("END OF RECEIVING DATA");
end TEST12;
```

```
-- Example 3

-- ------------------------------------------------------------------------
 -- UNIT_NAME          | TEST31.A
 -- UNIT_DESCRIPTION   | This program is designed to set baud-rate form 1200 - 9600 for the
--system which we want to use, the soft ware can easily expand by set more rate settings.
 -- AUTHOR             | CDR. Chin, Yu-Chi
 -- ADVISOR            | Dr. Uno R. Kodres
------------------------------------------------------------------------


with BIT, TEXT_IO;
use  BIT, TEXT_IO;
procedure TEST31 is
  CH  :  INTEGER;
  package INTEGER_INOUT is new INTEGER_IO(INTEGER);
  use INTEGER_INOUT;

    procedure TEST24(PORT1, PORT2, PORT3, PORT4 : in INTEGER) is
        NUMBER,
        SELECTION  :  INTEGER;
     begin
        OUTPORT (PORT1, 128);       -- access Baud-Rate- Divisor
        PUT_LINE ("SET THE SYSTEM BAUD-RATE AS YOUR DESIRE");
        PUT_LINE (" --- SELECTION IS --- ");
        PUT_LINE ("    1  =  1200      ");
        PUT_LINE ("    2  =  2400      ");
        PUT_LINE ("    3  =  4800      ");
        PUT_LINE ("    4  =  9600      ");
        NEW_LINE;
        GET(SELECTION);
     case SELECTION is
       when 1 =>
        NUMBER := 96;        -- Baud-Rate = 1200
        PUT_LINE ("YOUR SYSTEM BAUD-RATE IS SETTING TO 1200 NOW");
```

```
      when 2 =>
        NUMBER := 48;       -- Baud-Rate = 2400
        PUT_LINE ("YOUR SYSTEM BAUD-RATE IS SETTING TO 2400 NOW");
      when 3 =>
        NUMBER := 24;       -- Baud-Rate = 4800
        PUT_LINE ("YOUR SYSTEM BAUD-RATE IS SETTING TO 4800 NOW");
      when 4 =>
        NUMBER := 12;       -- Baud-Rate = 9600
        PUT_LINE ("YOUR SYSTEM BAUD-RATE IS SETTING TO 9600 NOW");
      when others =>
        PUT_LINE (" WRONG SELECTION !! TRY AGAIN !! ");
    end case;
      OUTPORT (PORT2,SYSTEM.BYTE(NUMBER));
-- convert the number into system.byte
 -- put LSB on 3f8 or 2f8 (com1 or com2)set Baud
      OUTPORT (PORT3, 00);      -- put MSB on 3f9 or 2f9
      OUTPORT (PORT1, 03);      -- no p one s 8 bit
      OUTPORT (PORT4, 03);      -- set modem control register
      OUTPORT (PORT3, 00);      -- interrupt control register (disable)
      NEW_LINE;
      PUT_LINE ("BAUD-RATE SETTING IS DONE !! ");
  end TEST24;

 begin
 NEW_LINE(5); SET_COL(5);
 PUT_LINE("This program is designed to help you setup the system BAUD_RATE");
 NEW_LINE;
 PUT_LINE("Make your choose for COM1 or COM2 first : use  -1-  or  -2-");
 GET(CH);
  case CH is
   when 1 =>
    TEST24(1019, 1016, 1017, 1020);
    PUT_LINE("FOR COM1 SETTING");
   when 2 =>
```

```
      TEST24(763, 760, 761, 764);
      PUT_LINE("FOR COM2 SETTING");
    when others =>
      PUT_LINE("WRONG KEY !! TRY AGAIN !!!");
  end case;
end TEST31;
```

```
-- Example 4
-------------------------------------------------------------------------
-- UNIT_NAME          | test35.a
-- UNIT_DESCRIPTION   | This program is designed for Suitcase Navigation
--                    | data logger which log data from GPS sensor store
--                    | it into a file and print the data string by the
--                    | user desire command
-- INPUTS             | GPS input raw data
-- OUTPUTS            | data record
-- CREATED            | 12 May 1991
-- AUTHOR             | Chin, Yu-Chi
-- ADVISOR            | Uno R. Kodres
 ------------------------------------------------------------------------
 with BIT, KEYBOARD, TEXT_IO, RTEXT_IO;
use  BIT, KEYBOARD, TEXT_IO, RTEXT_IO;
 procedure TEST35 is
 package INTEGER_INOUT is new INTEGER_IO(INTEGER);
 use INTEGER_INOUT;
 FILE, THE_FILE  : FILE_TYPE;
 ROCK            : INTEGER;
 ABORT_REQUEST   : EXCEPTION;
    -- A procedure to print the desire data_string
  procedure PRINT_DATA is
   ITEM  :  CHARACTER;
   FILE  :  FILE_TYPE;
  begin
   NEW_PAGE;
   NEW_LINE(5);
   PUT_LINE(" The current data_string is : ");
   NEW_LINE;
   OPEN (FILE, IN_FILE, NAME => "POS.OUT" );        -- open the update file
   while not END_OF_FILE(FILE) loop
     GET(FILE, ITEM);                               -- get the data
     PUT(ITEM);                                     -- put it on the screen
```

```
 end loop;
 NEW_LINE(10);
 CLOSE(FILE);                              -- line 140
end PRINT_DATA;
 -- A procedure as data logger
 procedure DATA_LOGGER  is
 A_CHAR       : CHARACTER;
 VALUE        : SYSTEM.BYTE;
-- A function to get data from GPS sensor
  function GET_CHAR return CHARACTER is
  LINE, DATA : SYSTEM.BYTE;
  LINE_INT   : INTEGER;
 begin
  loop
   INPORT (765 , LINE);            -- check 2fd input data is available
   exit when TSTBIT(INTEGER(LINE), 0); -- loop until it's available
  end loop;
  LINE_INT := INTEGER(LINE);           -- convert LINE into integer
  CLRBIT(LINE_INT, 0);                 -- clear TESTBIT
  INPORT (760, DATA);                  -- read data from 2f8
  OUTPORT(765, SYSTEM.BYTE(LINE_INT)); -- put value back to 2fd
  if INTEGER(DATA) > 127 then          -- a test loop for bad data
   DATA := 42;                         -- if not a character use '*'
  end if;
  return CHARACTER'VAL(DATA);
 end GET_CHAR;
 begin
 PUT_LINE("Waiting for the first legal character");
 loop                                  -- wait for the first legal char
  OPEN (FILE, OUT_FILE, NAME => "POS.OUT"); -- open a temp out file
  while GET_CHAR /= '[' loop           -- process each message received
   null;                               -- and wait for start character
  end loop;
  PUT ('[');                           -- put start char on the screen
```

```
    PUT (THE_FILE, '[');              -- put start char to file
    PUT (FILE, '[');                  -- put start cgar to temp file
    loop                              --
     A_CHAR := GET_CHAR;              -- get and put the message
     PUT (A_CHAR);                    -- put data on the screen
     PUT (THE_FILE, A_CHAR);          -- and to the file
     PUT (FILE, A_CHAR);              -- and to the temp file
     exit when A_CHAR = ']';          -- line  185
    end loop;                         -- get another set of data
    CLOSE(FILE);                      -- close temp file here
    exit when KEYPRESS;               -- exit loop
  end loop;
 end DATA_LOGGER;
begin
 OPEN (THE_FILE, OUT_FILE, NAME => "POS.DAT");-- make file pos.dat availabe
 loop
  if KEYPRESS(CURRENT_INPUT) then     -- press Space_Bar to interupt and
   NEW_LINE;                          -- ready to print the screen
   PUT_LINE("New data_string is as follow :");
   NEW_LINE;                          -- line 201
   ROCK  :=  KEY_VALUE;               -- assign keypress to variable
   if ROCK = 32 then                  -- if Space_Bar is true then
    PRINT_DATA;                       -- bring the data to the screen
   elsif ROCK = 81 or ROCK = 113 then  -- press (q) to exit the program
    raise ABORT_REQUEST;              -- raise exception    line 206
   end if;                            -- else program will keep running
  end if;                             -- no interuption
  DATA_LOGGER;                        -- program to log sensor data
 end loop;
 exception
  when ABORT_REQUEST =>               -- exit the program
 CLOSE (THE_FILE);
 PUT_LINE ("END OF PROGRAM TEST35.A");
end TEST35;
```

APPENDIX--V: USER's MANUAL (Operations and Responses)

O-1. Use RS-232 cable to connect the external Nav_Sensor and the Computer.

O-2. Connected the antenna and printer. Use Figure-9.

O-3. Start up the Nav_Sensor (GPS), and set the ASCII output mode.

-- Select POS (position data) as the main position data, alternative selection will be MEAS

-- (measurement data), do not select BINPOS or BINMEAS.

-- (Use Sensor's operation manual to obtain the function control)

O-4. Insert the disk and find out the working file (DATA_LOG.EXE).

O-5. key in DATA_LOG and RETURN.

R-1. Instructions to operate the program

"A brief guide to use this program:

1. Make sure your system Baud_Rate is matched

2. Run the procedure DATA_LOGGER at proper COM_PORT

3. Use Space_Bar and then Prt_Sc to get updated data_string

4. Use (q) or (Q) to exit the program"

R-2. Delay 5.0 seconds

"Give the COM_PORT number in (1) or (2)"

O-6. Key in your COM_PORT set up.

R-3. when 1 =>

"You've set Port1 = 1021, and Port2 = 1016"

when 2 =>

"You've set Port1 = 765, and Port2 = 760"

R-4. "Make your choice for COM1 or COM2: use -1- or -2-"

O-7. Again you want to set the Baud_Rate at the COM_PORT, Key in 1 or 2

-- In this program set 1200 is relatively better than the others

R-5. when 1 =>

FOR COM1 SETTING

when 2 =>

FOR COM2 SETTING

R-6. SET THE SYSTEM BAUD-RATE AS YOU DESIRE

--- SELECTION IS ---

1 = 1200

2 = 2400

3 = 4800

4 = 9600

O-8. Key in your selection

R-7. when 1 =>

YOUR SYSTEM BAUD-RATE IS SET TO 1200 NOW

when 2 =>

YOUR SYSTEM BAUD-RATE IS SET TO 2400 NOW

when 3 =>

YOUR SYSTEM BAUD-RATE IS SET TO 4800 NOW

when 4 =>

YOUR SYSTEM BAUD-RATE IS SET TO 9600 NOW

R-8. Waiting for the first legal character

-- At this stage the program is running a procedure named LOGGER and

-- the function GET_CHAR keep checking the line register to seek the

-- legal character "[" (user defined) then pull it in to the system.

O-9. By observation the screen displays, you should be able to identify the the transmittion rate is good or not, if not, make your adjustment.

O-10. Use (Q) or (q) to quit the program, and reset the Baud_Rate or you can use it any time when you want to quit the job.

R-9. END OF PROGRAM DATA_LOGGER.A

-- Warning ** When you quit the job, the raw data file will be renewed also. The best way is pull the data out to another disk then reset the system as you desire. (copy POS.DAT to A: POS0001.DAT)

O-11. Use SP (Space_Bar) to get the current position data_string, when you hit the SP :

R-10. The current data_string is :

[05 THU 143 23-MAY-91 00:09:44 36:35.6893N 121:52.5101W -0015 01.7 787236 +00 0.0 000.0 274.5 +3.0286E-11 4092 3 13,6,12,2]

O-12. Use print screen (Shift_PrtSc) to get the hardcopy if you wish to

R-11. Printer prints the screen status.

R-12. Program keep on surveying, displaying the data on the screen.

NOTES: Things you might want to know:

1. Normally the GPS is default its functions for its own benefits, we shall carefully follow the Sensor's manual to operate it.

2. Trimble-4000S needs to set up a reference position and it should be with in 0.5 degree.

3. Offset position should normally be zeros.

4. The position will drift if the PDOP is over 15.0.

5. The external computer may have difficulty keeping up with the data at high rates, and may lose characters. If all data rates are above 2400, then the data communication is unreliable.

6. There are two separate program TEST31.EXE, and TEST35.EXE which handle the COM_ PORT selection and BAUD_RATE setting separate from DATA_LOGGER, User can operate TEST31 to set up the connection, then run the program TEST35 to collect the navigation sensor data. (Source code exists in Appendix.4).

APPENDIX--VI: TEXT NOTES FROM GRADY BOOCH

EXCEPTION HANDLING AND LOW-LEVEL FEATURES

Where as assembly languages force us to work at the most primitive machine levels, high-order languages usually constrain us to work only at more abstract levels. Since programming in a high-order language is much more productive than in an assembly language, this is generally not a problem. However, we must sometimes refer to system-dependent features, such as the location of an input/output port or the representation of some data structure in memory. In the past, because high-order languages did not provide appropriate expressive power, we were forced to use a combination of high-order and assembly-language programs in solutions, an approach that complicated the solutions and hindered readability and maintainability.

In embedded computer systems, run-time reliability is also an important factor, (We are relating a satellite positioning system) there are sometimes exceptional situations beyond our control, such as hardware failures in peripheral devices or unexpected bursts of input data. We cannot predict when such situation might occur, but in a reliable system, they must be planned for. As a high-level language, Ada supports software development in both expressing low-level machine features and system-dependent features. In the best case, we would like the program to be able to respond to the exception, and continue processing with reduced capability. Ada permits us to write exception handlers to capture both predefined and user-defined exceptions.

[Reference Booch Chapter-17]

```
-- SHOW_ERROR :  exception;
-- When something happened;
-- raise SHOW_ERROR;
```

REPRESENTATION SPECIFICATION

As mentioned earlier, in embedded computer systems we must concern ourselves not only with run-time reliability, but we must often refer to machine-dependent characteristics as well. Here we are designing our own I/O package for a unique device and need to refer to I/O ports at a certain location in memory. The desire is not to step outside our language, Ada provides several constructs with which we may refer to such low-level features (Ada provides four classes of representation specifications, namely: length specification, numeration type representation, record type representation, address specification).

Representation specifications should be applied only for the purpose of efficiency, this system required for interfacing with external systems, Ada lets us create abstractions about them in high level terms. When we then use one of the features, we are actually writing a particular bit string, and this same feature may be used to express the assembly language mnemonics of our target machine. Note that we may not associate two representations in a single task to avoid program errors. But there may be cases at different times where two representations are more efficient or perhaps map our view of the world better. Because we want to reorganize the record to be used in another task, we spool a given record to be saved on a secondary storage device. Our solution would be to have one index searching representation and one storage representation, the program control the data effectively.

LIST OF REFERENCES

1. Dr. Uno R. Kodres, research proposal for "Small Navigation Data Logger Software System".

2. Edward Yourdon, "Modern Structure Analysis" 1989 by Prentice-Hall, Inc.

3. Grady Booch, Second Edition "Software Engineering with Ada", 1986 by The Benjamin/Cummings Publishing Company, Inc.

4. World Range Suitcase Navigation Feasibility Study, document dated 30 July, 1990, B. Cohenour, PMTC, RDD, Code 3144.

5. Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A-1983, Unite States Department of Defense.

6. Ian Sommerville and Ron Morrison, "Software Development with Ada" 1987, Assigned text for direct study.

7. Berzins and Luqi, "Software Engineering with Abstractions" 1991, by Addison-Wesley Publishing Company, Inc.

8. Trimble Navigation Model 4000SL/SLD GPS Surveyor Operation Manual.Trimble Navigation Ltd. 585 North Mary Avenue Sunnyvale, CA 94086 800-TRIMBLE.

9. Trimble model 4000 remote control RS-232 interface operation manual. April 1990 Revision D. Trimble Navigation, Ltd. 585 North Mary Avenue Sunnyvale, CA 94086-3642 800-TRIMBLE

10. IBM Personal Computer Hardware Reference Library, April, 1983.
International Business Machines Corporation P.O. Box 1328-W Boca Raton, Florida 33432

11. IntegrAda Version 4.2.1 Libraries.AETECH 380 Stevens Ave. Suite 212, Solana Beach, CA. 92075

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center — 2
   Cameron Station
   Alexandria, VA 221314

2. Dudley Knox Library — 2
   Code 0142
   Naval Postgraduate School
   Monterey, CA 93943

3. Director of Research Administration — 1
   Code 012
   Naval Postgraduate School
   Monterey, CA 93943

4. Chairman, Code CS — 2
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943

5. Prof. Uno R. Kodres, Code CS/Kr — 3
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA. 93943

6. AEGIS Modeling Laboratory, Code CS — 6
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA. 93943

7. Bernie Cohenour, Code 3334 — 3
   Pacific Missile Test Center
   Point Mugu, CA 93042-5000

8. RADM. Hung Cheng-Lo — 2
   P.O. Box 9617
   Washington, D.C. 20016

Thesis
C4447 Chin
c.1 The Navigation Data
Logger for a Suitcase
Navigation System.

Thesis
C4447 Chin
c.1 The Navigation Data
Logger for a Suitcase
Navigation System.

DUDLEY KNOX LIBRARY
3 2768 00016391 9